An Approach to Computer-based Knowledge Representation for the Business Environment using Empirical Modelling

by

Suwanna Rasmequan

Thesis

Submitted to The University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

Doctor of Philosophy

Department of Computer Science

University of Warwick

November 2001

To my parents

and

every member of the family

Table of Contents

List of Tables

List of Figures

Acknowledgements

First of all I would like to thank my supervisor, Steve Russ, for his advice, guidance and care to overcome the necessary difficulties of the process of research and writing this thesis. And a special thanks to his patience on those unnecessary difficulties arising from differences between our culture and language.

I would also like to thank Meurig Beynon and members of the Empirical Modelling Group at the University of Warwick to whom without their initiative and contributions this thesis would not have been written.

My thanks also goes to Tomkanok Jantarujirakorn who introduced and encouraged me to take the opportunity to complete my dream of doing PhD and to Tawatchai Iempairote and Seree Chinodom for their kind support. And to the Thai Government who sponsors the research.

Finally, my deepest gratitude goes to my parents and special thanks to my sisters, brothers, nieces, nephews and friends for their love and care that inspire this research.

* * * * * * * *

Further thanks goes to Malcolm Crowe for his constructive comments during the viva and to Steve Russ and Meurig Beynon for their useful help on this final version.

Declarations

This thesis is presented in accordance with the regulations for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree. The work in this thesis has been undertaken by myself except where otherwise stated. Significant ideas in this thesis have appeared in:

- S. Rasmequan, C. Roe, and S. Russ. Strategic decision support systems: an experience-based approach. In *Proc. of the 18th IASTED Conference on Applied Informatics*, 14-17 February, Innsbruck, Austria, 2000 [Rasmequan+00a].

- W. M. Beynon, S. Rasmequan, and S. Russ. The use of interactive situation models for the development of business solutions. In *Proc. of the Workshop on Business Information Research*, University of Rostock, Rostock, Germany, March 2000 [Beynon+00a].

- S. Rasmequan and S. Russ. Cognitive artefacts for decision support. In *Proc. of the 2000 IEEE International Conference on Systems, Man & Cybernetics*, 8-11 October, Tennessee, USA, 2000 [Rasmequan+00b].

A further paper, *'A New Paradigm for Computer-based Decision Support'* [Beynon+02] has been accepted for the Special Issue of the Decision Support Systems: the International Journal, on the theme *'New Directions for Decision Support Systems'*, which is forthcoming.

During the research work associated with this thesis, two other publications were written jointly by the author and other members of the Empirical Modelling group: [Beynon+00b, Beynon+01a].

Abstract

The motivation for the thesis arises from the difficulties experienced by business people who are non-programmers with the inflexibilities of conventional packages and tools for model-making. After a review of current business software an argument is made for the need for a new computing paradigm that would offer more support for the way that people actually experience their business activities. The Empirical Modelling (EM) approach is introduced as a broad theoretical and practical paradigm for computing that can be viewed as a far-reaching generalisation of the spreadsheet concept.

The concepts and principles of EM emphasise the experiential processes underlying familiar abstractions and by which we come to identify reliable components in everyday life and, in particular, business activities. The emphasis on experience and on interaction leads to the new claim that EM environments offer a framework for combining propositional, experiential and tacit knowledge in a way that is more accessible and supportive of cognitive processes than conventional computer-based modelling. It is proposed that such environments offer an alternative kind of knowledge representation. Turning to the implementation and development of systems, the difficulties inherent in conventional methods are discussed and then the practical aspects of EM, and its potential for system building, are outlined.

Finally, a more detailed study is made of Decision Support Systems and the ways in which the EM focus on experience, and knowledge through interaction, can contribute to the representation of qualitative aspects of business activities and their use in a more human-centred, but computer-supported, process of decision making. Illustrations of the practical application of EM methods to the requirements of a decision support environment are given by means of extracts from a number of existing EM models.

Abbreviations

AI – Artificial Intelligence

DEM – Distributed Empirical Modelling

DBMS – Data Base Management Systems

DE – Decision Explorer

DSS – Decision Support Systems

EM – Empirical Modelling

GDSS – Group Decision Support Systems

IS – Information Systems

IT – Information Technology

ISM – Interactive Situation Model

ISMs – Interactive Situation Models

KR – Knowledge Representation

OCI – One-way Closed Interaction

RMM – Restaurant Management Model

SDSS – Strategic Decision Support Systems

SSM – Soft Systems Methodology

SODA – Strategic Option Development and Analysis

TOI – Two-way Open-ended Interaction

Chapter 1

Introduction

1.1 Framework

This thesis identifies and makes explicit the special properties of the Empirical Modelling approach, based on observables, dependency and agency, that enable the construction of environments that incorporate a new form of 'knowledge representation'. This new form of knowledge representation makes an EM environment particularly suitable to support business system development. This kind of support is provided through the use of 'active' models, or 'cognitive artefacts' as they will be referred to later in this thesis. Active models are those models whose openness makes them actively reflect changes and movements in the business situation, in a way similar to how people use their mental models to make sense of the world and respond to it. For this reason they are well suited to describe business activities. In contrast, the term 'passive' model refers to here the product of computing paradigms which are based mainly on precise specifications or mathematical and logical structures. In deriving a mathematical or logical description of a domain for use in such a passive model, its components have to be abstracted and circumscribed. While this allows accurate prediction of the model behaviour, it means the model is detached from its context and is hard to revise or re-interpret.

This thesis is the first research work that explicitly relates knowledge and theories about knowledge with an Empirical Modelling approach and makes a claim that there is a new form of knowledge representation exhibited in the process of developing and using the models. EM knowledge representation is based on the close integration of human cognitive processes and computing processes which is a major difference between EM modelling and conventional software systems development.

Support systems for business activities

Most business decision makers are constantly dealing with volatile and human-centred real world situations. These situations are of both structured and unstructured types. The structured problem situations, as the name implies, are well-understood. Therefore certain solutions can be made available readily by structured problem solving or decision support systems. The unstructured decision problems, on the other hand, need a very good understanding of the situation to help with their solutions. This issue could be addressed by an unstructured decision support system if there was such software. It is obvious that no person can understand the problem situation as well as the person who has the problem. It may happen that an experienced person may understand better than a novice. Nevertheless, only the person who has the problem really knows what he or she is dealing with. Thus the computing environment that would allow the decision makers to develop their own decision models for unstructured problems might be a fascinating office technology for a current generation of e-business executives if it was available.

An example of such an environment that allows decision makers to develop their own business models is the business world's most popular office tool: the spreadsheet. Even the most recent versions of the spreadsheet are only suitable for modelling structured and semi-structured types of problem situation such as financial planning models. It is weak on handling anything more complicated than numerical data and it provides limited visualization of graph-based simulation (this is based on the use of spreadsheets in a 'normal' mode). Overcoming such limitations, if this could be achieved, would provide a better substitute environment.

Such decision support systems as can be produced by programmers often arrive too late to be an effective support for executives to make a correct decision. Or, they are too specialised and unsuited for all but a particular kind of problem. An environment which was more powerful than the spreadsheet environment and less complicated in both time and procedure in development than those obtained by programming paradigms could alleviate this situation. Decision makers, facing unstructured problems, might be able to enjoy creating their own decision support

models which they could implement by their own skill in problem solving via interactive experimenting with such an environment to solve their current problems. Such a model could be passed over to computer systems experts for further refinement if necessary.

Empirical Modelling for business activities

Empirical Modelling (EM) is a broad-based approach to computing that is both principled and practical. It is hard to make comparisons between EM and familiar subfields of computer science – such as software development or human-computer interaction – because the main focus of the methods and tools of EM begins with activities that normally precede the standard phases of programming. Such activities might be:

1. domain analysis - for example, establishing the relevant entities and mechanisms and the meaning and significance of terms and processes;
2. understanding and identification of problems and tasks; and
3. establishing relevant stakeholders in an enterprise and their initial viewpoints and requirements.

These activities are essential preliminaries to developing software solutions. For small tasks in well-understood areas they can, perhaps, be taken for granted. For large-scale tasks, however, or for tasks that are not yet well understood, they can raise long-term, difficult problems. In the conventional practice of programming, these activities are conducted, for example, by means of extensive listening and discussion, documentation, diagramming, natural language and specialist notations. While performing such activities, the thinking processes proceed mainly by imagination and mental modelling. Some of these methods may also be computer supported. What does not typically happen at this stage is to make rich, computer-based, interactive representations of a domain or system, where these representations can be directly experienced. Such rich representations, that offer direct experience, encourage active engagement and spontaneous involvement which, in turn, support cognitive processes. Building such representations is exactly the aim, in principle, of the approach

and tools of Empirical Modelling and it can, to a significant extent, be demonstrated in practice.

There is a qualitative difference in the medium for representation provided by EM from the usual means available such as mathematics, languages of various kinds, diagrams etc. This difference is more easily experienced than described. By the 'medium of representation' here is meant the combination of an interpreter supporting concepts of agency, dependency and observables together with some means for visualisation such as a windowing display system. The richness and openness of the medium shows itself in the unbounded way that layers of detail can be added to a model at any stage in its development or use. The medium is 'interactive' in the very strong sense of allowing experiments and engagement, in principle at least, with the same freedom that applies to the real-world referent. And the sheer speed of response for the user to see how a modification affects the model leads to an unusual sense of immediacy and directness of experience. In these ways – richness, openness and interactivity – we can compare an EM model with a physical model like a model ship in a wave tank. It is fundamental to the way an EM development proceeds that the modeller's experience of the model is comparable to (or of the same kind as) their experience of the referent. This is not the case when the modelling medium is largely symbolic [Russ97].

For example, if a room is being modelled in a geometric drawing program it is not surprising if a 'table' can be moved through the 'walls'. A geometric model is an abstraction, detached from its referent and only modelling those specific aspects for which it was intended. The 'room' model in EM is capable of subtle and experiential interactions. So that in spite of its appearance as a simple line drawing it will be argued in Chapter 3 that it has more in common with the painting of an artist than with geometric modelling. The crucial difference between such representations is that the former can always be enriched by adding further properties and relationships, the latter are circumscribed and limited by the initial decision about their types. The way we tell the difference between such kinds of representation is through interaction. The circumscribed, or abstract, representations have only a limited repertoire of possible interactions, the representations which are 'as-observed', or 'pseudo-physical', on the

other hand, can be endlessly enriched to respond to arbitrary interactions in imitation of their real world counterpart.

It is the aspiration of EM to use computers to build artefacts that are based on direct experience of parts of the world and that represent that experience in the above sense. This is an ambitious goal and much broader than the kind of modelling process that is typically undertaken with computers. It involves a major 'stepping back' from the starting point of conventional computing which begins with reliable components such as standard data types and algorithms, with object-oriented implementations and mathematical models. The major sacrifice in this move is the reliability of the components. In the everyday world of human affairs we cannot normally assume, without much experience or experiment, the reliability of entities, mechanisms or meanings.

Because EM begins from human perceptions and activities that are generally unreliable and seeks to make representations on computers that work best with reliable components it may be regarded as an approach to establishing the reliability of assumptions, mechanisms and meanings. This work of establishing reliability, or gaining confidence, can be considered in several ways. These are characterised by movements:

1. from state to behaviour. That is, to put more emphasis initially on the significance of state rather than behaviour. As a sequence of states leads to behaviour, the more we know about each state, the more we can have reliable patterns of behaviour to work with. In addition, the consideration of the states themselves can lead to alternative behaviours from what are currently perceived.

2. from experience to abstraction. That is, to move towards machine abstractions without losing contact with the richness of phenomena that is gained through experience.

3. from modelling to programming. That is, to begin thinking and modelling with the raw materials of experience rather than the types and conventions of a programming language.

4. from environment to system. That is, to entertain the many different aspects and viewpoints provided by a broad context, rather than limiting interest to the specific functions of a certain system.

5. from experiment to construction. That is, after gaining sufficient confidence in the components needed for an intended system and how they work together, through experimentation, the construction of a useful system may follow.

Note that the targets of these movements are all terms familiar in the world of computing and they can be thought of as bounding the automated computer processes that are appropriate for some application. But the goal of EM is not so much to take us from the unreliable to the reliable but to mediate between these worlds – the human and the computer-based – and embrace both of them. Hence an appropriate term for the broader framework of EM might be 'human computing'. Because it is explicitly bringing together the world of human processes and the world of computing processes it is likely to be particularly applicable to those application areas where there is the need for close integration of human and computer activity – areas such as business, design and learning.

It is in this everyday, muddled world that we begin with EM. Because of the uncertainty and complexity of such a world it is appropriate to focus initially on state rather than behaviour. State is presented to us as humans through what we can observe. There are 'chunks' of state that are coherent, or have integrity in themselves, that are naturally occurring (studied by science) or artificially made (by conventions, laws etc). Wherever such chunks of states exist then changes will typically occur respecting their coherence and so there are large areas of changes that can naturally be maintained by dependency. At the same time there are numerous changes of state that occur independently of others and are attributed to agents within the environment (such as humans, or gravity for example). Accordingly the fundamental concepts which guide our modelling processes are those of observables, dependency and agency.

The principles and philosophy of the Empirical Modelling (EM) Group founded by Dr. Meurig Beynon in the mid-1980s at the University of Warwick, UK form a new approach to systems modelling which has one basic principle in common with spreadsheet systems, that of 'dependency maintenance'. In addition, it emphasises real-world state representation and learning activities in which the user acquires knowledge through experiments with the system. This allows the construction of familiar and customised environments. Such alternative environments take seriously the incorporation of users' involvement during the system construction. There is such a need because of the fact that the long established tradition of information system development since the late 1950s [Hirschheim+95] has still not successfully captured users' requirements or delivered systems that are really useful to them.

The research in this thesis focuses on the application of the philosophy and techniques of Empirical Modelling to the challenge of building business DSS prototypes. The thesis argues for the position of EM as, *"... a candidate for computer-based modelling that is based upon a thesis about **learning and cognition** broad enough to embrace the role of the computer as an artefact and an instrument ..."* [Beynon97b].

Modelling for business system development

Software systems are generally large and require us to use standard programming languages, design notations and concepts that have traditionally been important in shaping people's thinking about system development. Structured programming, modularity, data types, algorithms, functions, objects and distributed systems are all part of what we shall term here 'conventional' programming, although the rate of change means such a concept will always be a moving target. Naturally such conventional methods have proved their worth and been successful mainly on well-defined systems. Human activity systems such as business processes[1] are far from being well-defined systems, so that a different approach is required in attempting to provide real-

1. A definition of the term 'business process' which is one of the controversial words used here based on the meaning given by M. Pidd is, "A process is a set of dynamic activities needed to get something done that will add value in the business." [Pidd96, p. 26]

istic support. Looking at another aspect of exploiting the use of computer-based systems, EM employs and enhances the common practice of modelling as a way of constructing a purposeful computer-based support system, in particular, of supporting business activities. The term 'modelling' employed by the EM approach encompasses not only its common meaning which, for most computer scientists, is the representation and manipulation of objects of study by the use of mathematical formulae, but also its associated meaning, as a representation of objects of study by the use of physical replicas, that is, visual forms that enable actual experience of the model which is normally found in the modelling process of creative product design or engineering design work. Beynon et al give an account of EM modelling activity as follows:

> *The EM approach offers a distinctive kind of modelling. With its focus on state, and lack of control structures, it is more primitive than conventional programming. State change is only achieved through automatically updated dependencies or the explicit action of an agent (who may be the modeller). With a conventional approach, the modeller must preconceive the processes involved. If there is a need for new inputs or outputs, e.g. an additional system feature is required, then the whole process may have to undergo costly revision and redesign. With the use of EM, revisions can in principle be made at any point during the modelling process with immediate effect. This is because the state of a model is captured in a script of definitions that resemble spreadsheet formulae. Each definition is either a value definition or a formula definition ...*
>
> [Beynon+02, p. 10]

EM researchers are not alone in giving such significance to the modelling process. Michael Pidd [Pidd96], writing from the perspective of management science, argues that if used sensibly, models and modelling approaches are 'tools for thinking'. That is to say, models and modelling processes are cognitive artefacts that assist human cognitive activity. He identifies different approaches in modelling e.g. soft systems methodology, cognitive mapping, qualitative system dynamics, optimisation of

spreadsheets, visual interactive modelling and meta-heuristic approaches. He classifies them into two categories: soft approaches and hard approaches (see § 2.4.2).

On the one hand, hard approaches in modelling are associated with a conventional way of using and developing computer systems. On the other hand, soft approaches are considered to be a radical change in use and development of computer-based systems. The consideration that it is necessary to make a change from the conventional practice in the way computer systems are used or developed to support human tasks is explained, for example, by Pidd's arguments that *"Mathematics and mathematical models are very useful ... But it is important to realise that the value of models and modelling approaches extends way beyond the realm of mathematical models for decision and control ..."* [Pidd96, p. 26]. In arguing for more emphasis on a soft approach, he also points out that,

> *... just as quantitative models may be used to demonstrate the effect of different policies, so other types of model may be used to explore the consequences of different ways of seeing the world. This is particularly important when operating at levels above the merely operational, where the issue is not so much how to do something, but more about what should be done. That is, there is more of a concern with ends than with means. In this strategic analysis it is quite normal for people to argue and debate from different presuppositions ...*
>
> [Pidd96, p. 120]

Modelling activity as introduced by the EM approach is a novel way of constructing computer-based support systems. We share Pidd's view that models and the modelling approach are capable of being tools for thinking. Pidd's example of the necessity of modelling the business process prior to business process re-engineering (BPR) to discover where the improvement of components will make a difference, gives a similar idea of how an alternative kind of modelling can provide an environment for constructing purposeful systems. This means seeking an environment that supports the finding of proper components of the system needed, instead of having an environ-

ment that offers another means to work on pre-defined components which may not match the real need. The essentials of such an environment are the *quality of interaction* and the *degree of flexible engagement* for the modeller. This is demonstrated by considering an EM model as a cognitive artefact that could enhance human thinking ability through learning, via open-ended interactions with models. Generally, EM models combine both abstraction-based computing and experienced-based computing. Abstraction-based computing is an approach that is generally used in most 'hard' computer-based systems i.e. conventional developed systems. Experience-based computing is an approach that is generally used in 'soft' computer-based systems such as spreadsheet systems. Another way to describe this broad approach to modelling in EM is to see it as a framework that includes both the modelling needed for software system development as well as the more informal, less structured modelling needed for thinking about systems and their requirements.

1.2 Motivation

The author's previous business experience in Thailand with different types of privately owned companies, suggests that the use of the computer in Thailand is limited to a certain group of people and that the computer's vast capacity is under-used. There is a very small number of both employee and management people who are involved with a company's computer system and its benefit. The computer system in this context refers to software, hardware, people-ware and so-called communication-ware.

The majority of people, regardless of their age or educational background, have been locked into a certain state of mind that might be called 'technology-phobia'. This type of person will always be afraid of getting involved with any of the so-called 'neo-tech' products. The computer system is counted as one of these products. The reason for this might be that apart from personal perception, it is only for the past decade that 'computer systems' have spread to small, medium and large scale businesses. However, the dramatic drop in hardware pricing as electronic technology has become more

advanced, has made the spread possible. In addition most of the hardware and software allow compatibility between each other. Even so the cost is still very high for some firms. For this reason, only a few people are allowed to interact with the computer system in a sophisticated way, either to store or to retrieve data. The rest of the staff may just make routine use of computers, for example, word processing. If this group of staff was allowed to access the company computer system fully, the company might benefit from the data or information which they could produce. However, there are always two sides to a question. A proper training programme has to be given to any new user and then followed with a user supported service – this costs money. Also there needs to be some control of sophisticated use, to protect company data.

The other possible reasons that obstruct people from making use of computer systems are the rapid development in computer technologies and the complication of the system itself. However, there is no way for any business to escape from the use of the computer system. It can be argued that day by day the introduction of computer systems is an increasingly necessary element to meet people's needs. In addition to the basic need for food, housing, medicine and clothes, many transactions of people's daily lives or business activities are carried out by the computer. People have been both directly and indirectly acquainted with the use of computer systems. For example, many electrical appliances in a house are controlled by computerized systems, for example, air-conditioning machines.

In the business area, rapid developments of Information Technology (IT), Information Systems and their subsequent systems or applications such as Management Information Systems (MIS), Management Support Systems (MSS), Executive Information Systems (EIS), Decision Support Systems (DSS), are the developments of current computer systems that can assist business people in solving more ambitious real-world problems than the earlier systems. Nevertheless, such developments emphasise the efficiency of the systems, in which case, give less attention to the effectiveness of the systems. This leads to the term **'no silver bullet'**[1]. That is to say there is no

1. This phrase is used by one of the most famous software engineering experts, Frederick P. Brooks in his important book, 'The Mythical Man-Month'. [Brooks75]

breakthrough software development solution yet for all real-world systems and this is likely to remain the case for some years to come.

My personal interest in computer technology (user-wise) and the problem of explaining proposals to programmers (in any language) made me consider researching the possibility of an environment that would allow:

1. a productive user interface: through which users themselves can learn and develop by interacting easily with the system;

2. an adaptive system: a system that is flexible enough to fulfil the changing nature of business practice; and

3. a knowledge generator: a system structure that permits powerful and flexible knowledge representation together with the possibility of gaining knowledge through system interaction.

This environment should represent real-world situations and a set of problem-solving tools for non-programmer users in such a way that users can learn and develop the prototype of the anticipated system with less expense and effort than by current means. The information era has introduced a dramatic change in the pattern of business conduct. 'Intellectual staff'[1] are now required in most organizations. The multinational, co-operation or joint venture culture no longer allows the advantage of one firm over another. In this situation, there is a new demand for knowledge as significant business capital for competence with the business concept: *"the right decision at the right time"*. The advanced developments in hardware technology have led to renewed optimism concerning the issue of **'Knowledge Representation'**, especially for business environments or other social science areas which have made very limited progress considering the effort made in recent years. The new nature of business conduct and the current distributed management strategy require a powerful remote decision support system.

1. Term used by S. M. Price to represent the staff who are well-educated in IT. [Price97]

The proposal for this research is to develop **"An Approach to Computer-based Knowledge Representation for the Business Environment using Empirical Modelling"**, aimed at finding a powerful alternative environment for decision makers in today's world of fast growing electronic transactions.

1.3 Theme

The aim of this thesis is to propose the application of Empirical Modelling to provide a familiar environment for business people to work with. Such an environment supports both the use and the development of computer systems for business activities that can be directly experienced. This familiar environment exhibits an integrated kind of knowledge and is a special kind of knowledge representation system. This thesis also attempts to answer the question of why existing computer-based support systems are often found to be alien artefacts to many people who do not have a formal background.

Since the beginning of the computer based era, there have been a number of different approaches in the computer world to try to capture the most valuable asset of mankind, namely knowledge. The most successful story in capturing knowledge might be those mathematical models that have been used for representation in the pure sciences. The ambiguous nature of social science has made the attempt to use the scientific method difficult if not impossible. This research study will be based on the theoretical standpoint that computer-based systems can assist in capturing knowledge in the business environment as well in scientific fields.

The original intention in this thesis was to have a case study that would show a construction of a DSS-prototype created by non-programmer users based on an EM environment. This 'ideal' environment would be an environment in which users with a moderate comprehension of computer systems (e.g. business executives) can easily learn to implement the prototype independently of a programmer. This is useful because the programmer may not have sufficient subject specific knowledge for the user's needs. In addition the way business users develop strategic thinking is more

intuitive and reactive than the systematic or logical way programmers develop their ideas. The idea is that there is an environment that allows business executives to develop the required DSS prototype themselves. There are two levels of use involved. Firstly, such an environment can be useful when the data involved is on a small scale and can be organised by the managers themselves. The environment would really be like an extended and generalised spreadsheet. Secondly, if an intended DSS involves a wide-range of data and can benefit more people, then the prototype can be a framework for developing a large-scale computer based support system.

This is based on our assumption that spreadsheet systems provide a good environment in quantitative modelling for quantitative aspects of business decisions. An EM environment incorporates spreadsheet principles and generalises them so they could then offer a better support for business people. That is, apart from supporting quantitative analysis, what is promising in an EM environment for business decision support, is the inclusion of qualitative aspects. The inclusion of qualitative aspects is made possible in an EM environment because we do not focus on any predetermined interpretation of interaction. The open-ended interaction allowed in an EM environment invites the negotiation of meaning which is typically needed for qualitative aspects.

The inclusion of qualitative aspects can help with two kinds of strategic decision support. First, decision makers can use an EM environment to model a problem situation. Such modelling may help when it is not clear what is going on in a situation. The very process of modelling may help the manager gain a better understanding of the situation which may itself resolve the problem. Also the power of modelling offered by computer systems can help provide a solution for such a problem if it was not already resolved during the process of trying to understand it. Second, as an EM environment supports experience-based modelling where the human is an integral part of the modelling process, it is possible for the models built by managers to represent knowledge by acquaintance in addition to descriptive knowledge (i.e. propositional knowledge). There are two kinds of knowledge by acquaintance (see § 4.3) that co-exist with propositional knowledge in an EM environment. They are tacit knowl-

edge and experiential knowledge. These two kinds of knowledge are what generally make the major contribution to the cognitive difference between people. In this thesis, we claim that we can include these two kinds of knowledge in our environment, therefore an EM built DSS can be used to support firms to have better 'capabilities' than if they were gaining support from conventionally built DSS.

The research needed to provide background and arguments for the above hypothesis took longer than initially expected and this has not permitted time for the development of such a prototype to be included in this thesis. In the discussion of future work (see § 6.4), a preliminary study of Soft System Methodology (SSM), which is a supporting approach in developing user-developed DSS as an example of business system development, is presented. Nevertheless, the author's involvement in the construction of the Timetabling Model (Temposcope) and the Restaurant Management Model (RMM), together with the study of existing models and their modifications, allows illustration of the key elements of a DSS-prototype, and the qualities that an EM approach can bring to those elements. Where possible, this thesis has included such illustrations by means of scripts, or screenshots, of EM models already developed.

It should be emphasised that we have little experience of 'system building' – neither the author, nor the group as a whole – although very many artefacts have been built. This thesis describes prospects for the application of EM to business and justifies why it is worth the effort of exploring this application. This research makes it clear that there is a special suitability of EM in terms of identifying and fashioning reliable components, embodying experience and being more human-centred than other paradigms.

1.4 Outline

In chapter 1, the chapter begins with the framework of this thesis. Then in the following section motivation of this thesis is discussed, followed by the theme and thesis outline. In the last section, thesis contributions are presented.

In chapter 2, a review of computer-based support systems for business activities is presented, in terms of existing tools that support business activities and issues around such support systems in relation to the actual needs of business activities. This thesis focuses on Decision Support Systems among other business support systems. This is on account of the author's own experience in firms she has worked with that it is actually bad practice that decision making processes rely so much on specialists in the Electronic Data Processing Department or Information Technology Department to supply the data and information, or decision support models. Firstly, such activity should be done by the people who want to use the information, as they know best what they want and in what way they want it. Besides, the most important issue here is that they should be able to access it at any time they want, not periodically as normally presumed. Secondly, support for strategic decision making is assumed to be restricted to only a few key members of staff, not to be spread among members of staff in those support departments. The obvious observation that can be made in this situation is that the interaction between the computer-based systems and users seems to require special computer skill that is unfamiliar to most managers. These issues lead to questions about whether the current main usage of computing for business activities is the only way to use computer-based support systems.

Discussion of knowledge-based systems, hard and soft approaches, and spreadsheets and their limitations, leads to an argument for the need for an alternative environment for business support systems.

In chapter 3, following from the discussion of chapter 2 on the need for an alternative environment, a proposal that there is a need for a shift in computing paradigm is presented. The Empirical Modelling approach is introduced as a shift in computing paradigm for supporting business activities.

In chapter 4, the main theme is the significance of the Empirical Modelling approach for a special kind of knowledge representation. This is identified and developed for the first time. It is proposed that through this form of knowledge representation Empirical Modelling can be used to offer strong cognitive support for business solutions.

In chapter 5, an overview of software systems development is given and an alternative approach to system development in the framework of EM is outlined. These discussions aim to put into perspective the inherent difficulties, e.g. changing of requirements and lack of end-user computing for business-oriented system development arising from contemporary approaches such as object-oriented technology. Then there is detailed discussion of the design of DSS and how the EM approach to knowledge representation and system development is particularly applicable to the needs of DSS. This is illustrated by showing how various technical requirements of DSS have already been addressed to advantage in existing EM models.

In Chapter 6, conclusions are drawn from the main chapters, an assessment is made of the research and there are proposals for future work.

1.5 Thesis Contributions

This thesis proposes an alternative approach to computer modelling for business support which possesses principles which are thoroughly grounded in human experience together with practical tools. There are three main areas of contribution: for the business needs for computer-based support, the EM approach itself and for the wider computing community.

Business needs for computer-based support

This thesis identifies the limitations of current computer support systems and proposes an alternative approach that addresses some of those properties that are weak in conventional approaches. The main issues in this area are:

1. The stereotype of conventional use is focused on optimising the power of computers. To achieve that two major features are sacrificed: i) The richness of phenomena in terms of observables and state-as-experienced with the use of abstract data types in describing state-as-abstracted. Observables, and state-as-experienced, refer to characteristics by which people make sense of the world. ii) Human involvement during the data manipulation process by having machines running as automatically as possible. Human involvement here refers to human capabilities, e.g. discrimination among choices, perceiving non-symbolic patterns etc.

Because of these limitations, and the pre-defined types of interaction allowed in conventional approaches, such stereotypes are not suitable for the use and development of decision support systems where the problems or situations are not clear to problem solvers or decision makers. An example of such a system would be a strategic decision support system (SDSS).

Open-ended interaction and the emphasis on modelling instead of programming of an EM environment in developing a system can address those limitations. This is because in an EM modelling process humans play an active role in shaping the data manipulation process to fit a purpose (that is, there is an integration of soft and hard aspects in an EM modelling process). It is not possible to write a program to support a decision process when there is no procedure to derive solutions. Besides, the conventional approaches to modelling with an emphasis on data manipulation alone do not correspond with the needs of this kind of system.

2. Building SDSS demands business person experience to configure models which need support from computer-based systems. The conventional approaches that put more emphasis on automation by emphasising abstraction

prohibit the incorporation of the expertise of business people who are not usually computer experts. This may be one of the reasons that actual strategic decision support systems have never been created. However, the beginning of a shift in computing paradigm by spreadsheet systems introduced a certain degree of change in computer use and development for end-user computing. Nevertheless, spreadsheet systems were limited by a small number of data types, built-in functions and its highly structured design (properties commonly found in conventional approaches).

In conclusion, support systems are hard to modify and adapt, even slightly, especially by end-users wishing to incorporate new experience. These limitations derive from the paradigm for computing itself. We have inherited this computing paradigm and it is reinforced by commercial pressures. That is, the limitations are not simply due to lack of technology, they are inherent in the paradigm. They are particularly apparent in business support.

The Empirical Modelling approach

This thesis is the first substantial work that applies EM to business and it proposes Empirical Modelling as an alternative framework for business solutions. This alternative calls for a shift from viewing the computer as algorithm machine, or even information-processing machine, to something like a *modelling instrument* that needs to be learned and used by human agents.

This thesis provides concrete theoretical arguments for special properties of EM models as follows:

1. they are experience-based as compared to abstraction-based,
2. they are cognitive artefacts that can extend the limits of the human cognitive process,
3. they can be used as instruments rather than as tools, and
4. they offer a special kind of knowledge representation.

The first three properties have been noted in a number of other EM research works but none of those works has provided fundamental arguments for their validity and significance or applied them to the issues associated with knowledge. The fourth property is newly identified and analysed in this thesis.

This thesis draws attention to an interesting feature of an EM environment, that is, there is an integration of propositional knowledge, tacit knowledge and experiential knowledge. Tacit and experiential knowledge are two kinds of knowledge that make one person different from another and hence the inclusion of these kinds of knowledge in the support systems can make a business problem solving process or decision making process of one firm different from another.

The third area that this thesis has made contributions to is the computing community.

The wider computing community

In respect to the computing community, this thesis has drawn attention to, and is able to address two main issues:

1. the potential for incorporating qualitative aspects into computer-based systems by using an EM approach and its environment. This inclusion is possible in an EM environment because of the integrated kind of knowledge that exists during the use or development processes through open-ended interaction between people and computers and the focus of an EM approach on the significance of state-as-experienced.

2. the orientation of an EM environment towards end-user computing. That is, the experience-based modelling of EM supports the construction of systems for individual purposes. That is to say, the incremental building of models within an EM environment develops from the growing understanding of users or developers during interaction with each component of models in tune with the interaction that they can make with the referent. At the same time their interaction with the real world referent shapes the structure of the models to serve the given purpose.

Chapter 2

Review of Computer-based Support Systems for Business Activities

2.1 Introduction

The two arenas of the computer world, the hardware and the software, have been growing with very different rates of success. That is, the success story of the invention of transistors to replace vacuum tubes in the early days of computer machines, and the subsequent development of integrated circuits and cheap memory, have made a dramatic contribution to today's computer hardware evolution. On the other hand, the story of the evolution of the software is not so spectacular. There are still regular reports of software projects that are late, over-budget or that do not deliver the services required. One of the reasons might be that the practice of software engineering is still in its infancy; it is the work of artisans or craftsmen and lacks the theory of a mature engineering discipline [Xia98]. This suggestion might appear surprising as it seems to contradict the general understanding that computer software is highly structured and 'scientific'. Another consideration is that in spite of the term 'software engineering' it may still turn out that software is not well suited to be an engineering discipline at all.

Hardware development is a consequence of the application of well-tested results and theories of physics and engineering whereas software methodologies are governed to a large extent by pragmatic and psychological considerations which are not subject to the application of such scientific methods. For example, *"... many software engineering methods seem to be useful, but usefulness should not be the canon for accepting theory that has no understanding power..."* and further, *"... software development is a cogni-*

tive process, without a good understanding of program(s), we can neither test nor maintain programs ..." [Xia98, p. 62-65].

There are many experienced practitioners who believe that software engineering is still far from being understood in the sense of engineering or science. F. P. Brooks in "The Mythical Man-Month" [Brooks75] is well known for drawing attention to the paradox that increasing the number of personnel on a software project can actually decrease the outcome of the project over a given period of time. Even in its recent edition in 1995, Brooks still argues strongly that software development is not well understood. Besides, software design is like any other creative activity in being a trial and error task, although, based on systematic procedures, it has some similarity with making scientific experiments. Both scientific discovery and creative activity are highly dependent on human faculties such as experience, interpretation and imagination. Each step is initiated, thought about and judged carefully. Without human faculties this could not happen. In software design there is an essential combination of scientific and creative activities. So far, systems have largely been designed with relatively little user involvement. There might be much more scope for taking advantage of user capabilities. H. V. Vliet's assertion supports this argument, *"Rather than approaching system design from the point of view that human weaknesses need to be compensated for, we may take a different stand and consider computerized systems as a means to support human strengths ..."* [Vliet93, p. 172].

The above discussions point to the issue that the construction of computer-based support systems, and in particular, the design of software, needs to pay attention to the value of making use, as much as possible, of human faculties in conjunction with the computer processes. If this is the case, then it is questionable whether current approaches to software design provide a good basis for such work. This leads to a further question whether programming principles, which impose the abstractions of type (or object) and function, are the most appropriate as an environment for business software design and construction. This thesis attempts to respond to these questions.

This chapter contributes a brief analysis of 'computer-based support systems' for business activities. The analysis aims at identifying the difficulties that obstruct advances in both the use and the development of software systems for business activities. At the end of the chapter, an alternative environment for business support systems is proposed.

2.2 Defining the Term 'Computer-based Support Systems' for Business Activities

The term 'Computer-based Support Systems' for business activities refers to the use of computer systems to assist people in their working processes. In general, computer-based support systems are used to support business people in performing their tasks by ways of exercising planning, managing, and control over business activities more effectively. Computerised systems can assist in that they can reduce people's efforts in calculating, manipulating and organising data and producing reports. The familiar uses are, for example, preparing accounting and financial reports, production control and scheduling, inventory control and distribution planning. From this perspective, the purpose of introducing a computer-based support system is mainly to facilitate the data collection process of the firm's activities as a whole, in addition to facilitating each process. The ultimate goal of data collection is to be able to have control over every activity within the firm. This data collection process with the use of modern digital computers provides rapid feedback about the company status in a particular area of interest, which in a pre-computer era would take hours, days, or months to find out.

So far, business computer-based support systems are oriented more to one side of the coin: to reduce cost and increase speed and accuracy of the work processes, or in broader terms, to use computer-based support systems as tools. Computerised tools can be designed to achieve many purposes and to involve little effort by the user. This is based on a fixed boundary structure of the device where most of the components are processed automatically. Examples of such tools are photocopying machines and

washing machines. This thesis will explore the other side of the coin: using computer-based support systems as instruments. With the use of instruments, people are integral to the accomplishment of a task, whereas tools will only perform the task for which they are designed. The skill and experience of the users of the instruments account for the success or failure of the tasks. For example, musical and medical instruments involve a great deal of users' skill and experience to fulfil their performance and task assignment respectively. Further discussion of the contrast between tool use and instrument use is in § 3.4.2.

When first introduced, spreadsheets served as tools for business people. Developments in spreadsheets that allowed more sophisticated uses by incorporating more human intervention shaped their mode of use to move towards that of instruments. This reflects a concern for how people should benefit better from computer-based support systems that is neither recent, nor only to be found in the work of this thesis. For example, back in 1968, Boutell [Boutell68] sees the value of human integration with computers. He illustrates those two sides of the coin when discussing the need to pay attention to the use of computer-based systems as a central device rather than a peripheral device. Boutell argues for a computer-based system structure that allows human intervention, in contrast to a batch-controlled system that limits human intervention. A system that allows human intervention is considered by him as a means for using the computer-based system as a central device. Human intervention as referred to by Boutell is, for example, the incorporation of live management criteria into the system components of certain business processes while these are occurring.

2.3 Current Usage of Business Support Systems

2.3.1 Usage in terms of types of support

The study of this research shows that the current usage of computer-based support systems for business can be classified in broad terms according to types of support as follows:

1. Systems to process large volumes of data that need the same processing or duplicated effort involving, for example, a company payroll system, a sales invoicing system, and a mail-order processing system.

2. Systems to process data from different locations, for example, a holiday booking system, an airlines alliance system, and an internationally-based company customer accounts system.

3. Systems to process very accurate calculations, for example, a commercial aeroplane navigation system, a production process control system, and a computerised medical treatment equipment control system.

4. Systems to process any data that, if it was done manually would take a longer time and be inefficient, for example, a stock control system, a problem simulation system, and a numerical weather forecasting system.

5. Systems to process data that needs to be constantly updated and accessible, for example, a credit card user data system, a customer bank accounts system, and a stock-market system.

6. Systems to process electronic commerce, for example, a digital catalogue system, an on-line booking and selling system, and an on-line banking system.

However the above classification is not intended to give a unique category for types of computer-based support. The categories are not disjoint. For example, a system to operate a cash machine (ATM) falls into both a system to process data that needs to be constantly updated and accessible and a system to process data from different locations. A supply chain system needs to process a large volume of data and to process data from different locations and be constantly updated and accessible. These

two systems, that is a cash machine system and a supply chain system, can also be given another classification as follows which classifies systems according to how they are used in respect of time as real-time systems.

2.3.2 Usage as classified by tools

Contemporary business uses of computer-based systems are focused on 'systems' that process the required information. These tools are known as 'Information Systems' (IS). These are, for example, 'Management Information Systems' (MIS) and 'Executive Information Systems' (EIS). Later developments changed the focus of information systems, and enabling systems were developed. Examples of these are 'Decision Support Systems' (DSS) and 'Expert Systems' (ES). D. Flynn [Flynn98] identifies and classifies IS according to the use into: 'transaction processing systems', 'decision support systems', 'real-time systems', 'database systems', 'expert or knowledge-based systems' and 'office information systems'. All of these, regardless of what they are called, support management at all levels of decision making and problem solving activities.

Transaction Processing Systems. Examples of transaction processing systems are: bank withdrawals or deposits; theatre seat reservations and patient appointments with a doctor. Transaction processing systems in general assist functional activities and usually contain additional procedures that can show the basic functional information to management. Such systems then are named 'Management Information Systems'. Flynn states that such additional procedures are quite straightforward and perform simple transformations such as aggregating the data, sorting it into different sequences or averaging.

Decision Support Systems. 'Decision Support Systems' (DSS) are one type of 'Information System'. This is well in line with the comment from Steven Alter, who is among the first group of people to introduce the term and its use.

> *Many of the basic ideas are still applicable in some way, but much of what was new about DSS at the time has now been incorporated into everyday use of office software. In my own work I have come to believe that the distinction between DSS and other types of information systems is much less useful today: Current DSS are often part of a larger information system. And the operation and significance of that larger information system can be understood best by looking at its role in specific work systems, such as the way people schedule a factory or the way people find new sales prospects ...*
>
> [Personal e-mail, 1998]

Such systems contain computer processes that typically apply mathematical techniques, particularly algebraic or statistical, to the analysis of information and the generation of solutions to problems to facilitate decision-making in management activities. Flynn explains that such systems are often built on Simon's model of decision-making process. He suggests that systems which assist middle management are generally termed 'Decision Support Systems', while those which assist top management are termed 'Executive' or 'Enterprise' Information Systems.

Sauter [Sauter97] suggests that 'Decision Support Systems' occupy a distinct position within the plethora of computer-based tools being developed in the areas of 'Information Systems' and 'Expert Systems'. For example, they offer more sophisticated support in terms of the generation and analysis of alternative solutions to a problem than can be expected from 'Management Information Systems' and are not as specialised as 'Executive Information Systems'.

'Executive Information Systems' and 'Decision Support Systems' are, according to Flynn, computer-based systems which analyse data against 'expert' criteria and then produce a list of alternative choices for a human expert to make a final decision which is based on intuition and has a 'seat-of-the-pants' character. He argues that DSS and EIS are quite difficult to design as they involve the integration of human decision-making activities, as compared to transaction processing systems that are designed to

monitor daily operations only. Although Alter (in the quotation above) has suggested DSS are becoming 'buried' within larger IS there is evidence for the opposite tendency. This is because of the recognition of the need to retain and integrate human factors in the decision making process. For example, a consultancy on software development (Software Engineering Decision Support Laboratory, University of Calgary in Canada – see www. seng-decisionsupport.ucalgary.ca) is deliberately emphasising the need for support systems for the decision making needed in the software development process.

Real-Time Systems. According to Burns and Wellings [Burns+90], real-time or embedded systems are systems where the prime function is not that of information processing. Nevertheless these systems require information processing in order to carry out their prime function. Such systems, *"... place particular requirements on the computer languages needed to program them – as they have different characteristics from the more traditional information processing systems ..."*. They further state that *"... the correctness of a real-time system depends not only on the logical result of the computation, but also on the time at which the results are produced ..."*. They identify two types of real-time systems. The first one is the 'hard real-time systems' where the response times are the critical factors of the system's failure or success. The second type is the 'soft real-time systems' which still function well if their response times miss the target. Soft real-time systems, according to Burns and Wellings, differ from interactive systems in the sense that interactive systems have no explicit deadlines.

Database Systems. This is a type of 'Information System' which is an integrated collection of files for recording information, together with software to provide user-oriented interfaces for fast access. Database Systems store an organisation's functional data centrally so that this data can be used simultaneously by several other information applications.

In 1992, Avison [Avison92] presents a 'database approach' as an appropriate method to construct organisations' computerised systems. This approach, he suggests, can separate data resources and their application systems (data independence).

Such separation enables flexibility when changes occur. For example, if the functions are changed, the data on the database will probably still be appropriate. Likewise, if the facts are changed then the database can be amended without redesigning the application systems.

Expert Systems. 'Knowledge-based Systems' is used by Flynn as a synonym for 'Expert Systems'. He states, *"... knowledge-based systems (or expert systems) share similarities with decision support systems in that they are not based on transactions, although they may use transaction data ..."*. Expert or knowledge-based systems refer generally to systems which store facts of the domain and have an inference engine to derive certain facts corresponding to criteria via the interfaces for making such queries. The term 'Knowledge-based Systems' is also used in specific senses, such as 'Knowledge-based Decision Support Systems', as in the work of Klein and Methlie [Klein+95].

Office Information Systems. Flynn gives the name Office Information Systems to: traditional paper-based activities found in offices, word processing, 'e-mail', fax messages, telephone answering and message recording, document management, personal diaries, meetings and conferences. An example of a tool of this kind is a 'groupware' system which provides a shared, text-oriented database, and e-mail subsystems for team members to communicate with each other.

2.4 Development of DSS

DSS is one of the main areas of computer-based support systems. The related kinds of information studies, decision analysis, decision modelling, operational research and management science, have developed alongside DSS in a complex pattern over the past three decades. While the computer hardware has been totally transformed over this period there has been only very slow progress in software to harness computer power and link it smoothly with human needs and cognition. For this reason much of the history of such software in business applications remains relevant today. This is especially true for this thesis where we are addressing problems that have changed little in nature since they first presented themselves in the 1960s and we are proposing a

shift in the computing paradigm that has shaped much of our thinking about technical support for these problems since that time. We now give a summary of this development and the associated literature on which we have drawn under the headings: history, current issues and future trends.

2.4.1 History

The book 'Theory of games and economic behavior' (1944) by John von Neumann and Oskar Morgenstern [Neumann+53] marks the early emergence of decision theory (see 'History of Decision Making' research report in [ArlingtonWeb]). James Cortada [Cortada96] states that the first occurrence of the term DSS is in the article "A Framework for Management Information System" written by G. A. Gorry and M. S. Morton, Sloan Management Review 13 (Fall 1971):55 -70. The above mentioned research report by Arlington Software Corporation supports this claim, it states, *"The advent of modern computer systems and large legacy databases from the late nineteen sixties onward has also served as a starting point for new challenges in the organizational decision making process ...".*

According to Daniel Power: *"The concept of Decision Support has evolved from two main areas of research: the theoretical studies of organizational decision making done at the Carnegie Institute of Technology during the late 1950s and early '60s and the technical work on interactive computer systems, mainly carried out at the Massachusetts Institute of Technology in the 1960s. (cf. Addison Wesley series on Decision Support forwarded by Peter G.W. Keen and Charles B. Stabell, May 1978) ..."* [PowerWeb].

Definition and formation of DSS. The following are a summary of DSS definitions from the work done by Efraim Turban [Turban95]:

1. 1970 Little (a refinement of Gorry and Scott Morton in the early 1970s work): *"a model-based set of procedures for processing data and judgements to assist a manager in his decision making"*

2. 1971 Scott Morton: *"Interactive computer-based systems, which help decision makers utilize data and models to solve unstructured problems"*

3. 1978 Keen and Scott Morton: *"Decision support systems couple the intellectual resources of individuals with the capabilities of the computer to improve the quality of decisions. It is a computer-based support system for management decision makers who deal with semi-structured problems"*

4. 1980 Steven Alter: Alter has clarified the concept of DSS by contrasting it with traditional EDP (Electronic Data Processing) systems on the following five dimensions.

TABLE 1. DSS versus EDP after S. Alter [Alter80]

Aspect	DSS	EDP
use	active	passive
user	line and staff management	clerical
goal	effectiveness	mechanical efficiency
time horizon	present and future	past
objective	flexibility	consistency

In spite of these definitions, Turban states, *"It should be noted that DSS, like MIS and other MSS*[1] *technologies, is a content-free expression (i.e. it means different things to different people). There is no universally accepted definition of DSS ..."*. The definitions of DSS available in the literature share one concept. That is, any DSS which is computer-based is created to support decision makers facing complicated problems by using software tools to inform choices for final decisions. The most recent and comprehensive definition is given by V. Sauter [Sauter97] as follows:

> *Decision Support Systems are computer-based systems that bring together information from a variety of sources, assist in the organization and analysis of information, and facilitate the evaluation of assumptions underlying the use of specific models. In other words, these systems allow decision makers to access relevant data across the organization as they need it to make choices. The DSS allow decision makers to analyze data generated from transaction processing systems and other internal information sources easily. In addi-*

1. Management Support Systems: term used by F. Turban [Turban95]

> *tion, DSS allow access to information from outside the organization. Finally, DSS allow decision makers the ability to analyze the information in a manner that will be helpful to that particular decision and will provide that support interactively.*
>
> [Sauter97]

The above definitions taken together, emphasise three properties of DSS: they are model-based, interactive and involve the human subject in tackling semi-structured or unstructured problems. However, as far as the current paradigms of computing are concerned the extent of those properties is still limited and the character of problems that they can tackle is still largely confined to the structured type of problems.

The Concept of DSS. Unless otherwise indicated, for the remainder of this section we are drawing mostly on the work by Klein and Methlie [Klein+95] in summarising DSS background and development.

Systems for decision support are clearly intended to be practical. They also have a strong theoretical or conceptual basis. As early as 1960 Herbert Simon had proposed that decision making can be considered in terms of three processes: exploration, analysis and evaluation [Simon60]. These phases, which are not necessarily sequential, of finding out exactly what the problem is, developing the possibilities for its solution and then, after considering the consequences of each, deciding among those possibilities, have become the defining trio of characteristics of the decision support process. Simon later adopted the language of intelligence-design-choice, while Mintzberg adopted identification-development-selection [Mintzberg73]. From this time, the potential of computer support, through programming of some kind, was being discussed. But there has been little guidance or illustration on how to conduct, or to support, the initial phase of problem identification which is perhaps the most challenging phase of the three.

Since these formative ideas about decision support expressed the process in terms of 'problems' it is not surprising that decision support has always been closely associated with 'problem solving'. Klein and Methlie treat these terms as synonymous and

that practice is followed here. However, the major concern in this thesis is that models of, or support for, these activities should take account of the personal character of the individual performing the processes. Therefore there will be inherent limitations in generic systems and end-user development and interaction will have a high priority.

Cognitive Aspects of DSS. The 1950s and 1960s saw a radical shift in perspective in the psychology of human behaviour. Instead of the 'black box' view of the mind endorsed by behaviourism, the work of Miller, Chomsky and of Newell and Simon (among others) focused attention on the internal workings and mechanisms in processes such as memory, language learning and problem solving. These became major topics for the emerging subject of cognitive psychology which began to compare human mental activity with the operation of computers and programs. There developed the theory that when performing problem solving humans employ 'information processing'. Newell and Simon introduced the concepts of a problem space (consisting of knowledge states) and heuristic search (rules of thumb for searching a model of the problem space). The essential technical idea in this theory was that human behaviours were controlled by programs in the form of production systems. These had three major components: memory for symbol structures representing the problem space, memory for the production rules in the form of condition-action pairs and a control strategy of matching, conflict resolution and action. These production systems held the promise of being more flexible and adaptive than procedural programming and being more suited to ill-structured problems.

The human problem solving process described above, that the human subject possesses a mental image of a problem and then proceeds with a heuristic search for a solution or solutions, does not seem quite right for all circumstances. It sounds as though we could have a complete knowledge of what is going on in that problem or that difficult situation. In most cases, we do not. That is, if we have a clear image of what the problem is then actually we can proceed with actions to solve such problems. The problem is, most often, we don't quite know what the problem is. Therefore we need some support to study the situation and to find out about the problem and then to proceed with necessary actions.

Production systems assume a fixed pattern of condition-action pairs and rule governed behaviour in response to human actions. From an Empirical Modelling perspective, those fixed patterns of condition-action pairs and rules governing behaviour can be used to model only reliable phenomena e.g. the law of gravity, cash flow analysis and satellite bandwidth allocation. Such a kind of system structure may not suit those phenomena that have no known reliable pattern of condition-action pair e.g. consumers' demand, customer behaviour and stakeholder satisfaction.

Normative and Descriptive Approaches. The 'normative' approach to decision making seeks to find the optimal choice – one that will maximise usefulness. Such an approach, with roots in economics and logic, is associated with formal models and underlies operational research. It will work well only in situations that are relatively well structured. The 'descriptive' approach looks instead at what actually happens in human decision making. This uses Simon's intelligence-design-choice principle and his notions of bounded rationality and 'satisficing' – the idea that people satisfy a problem requirement rather than optimise the solution. Such an approach makes much use of cognitive theories, for example, that problem solving proceeds by heuristic search and that this can be described by production rules and implemented in expert systems.

To cope with the real-world complexity, both 'normative' and 'descriptive' theories are needed and considered to be the basis for designing computer support systems. Klein and Methlie claim that 'Knowledge-based Decision Support Systems' as proposed by them can combine the two technologies in computer-aided decision making that is a decision support system which focuses on data access and analytical modeling and where knowledge representation and reasoning are the major concerns.

In respect to the discussion above an Empirical Modelling approach is more concerned with the descriptive approach for decision support systems. But the method employed, that is, a special kind of modelling process has some shared principles with the modelling method in the normative approach. However, there is a fundamental difference in the philosophy and tools that support the construction of an EM type of

decision support system. There is not so much interest in models to test cognitive theories of decision behaviour, but there is more concern with how a computer-based system using observables and agency can support human decision making. That is to say, an EM style of DSS allows human knowledge to grow and the user to gain insight through interaction.

2.4.2 Current issues

The hybrid nature of information systems and the state of the art of decision support systems operating over the internet or local intranets [Bhargava+95a] are two current issues that bring in new ways of thought for DSS developers. The hybrid nature of information systems in particular suggests that old distinctions (e.g. between MIS, EIS and DSS) are breaking down and that some components of DSS are likely to be included in any information system. As Alter says, *"Current DSS are often part of a larger information system ..."* and *"DSS has been now incorporated into everyday use of office software ... "* [personal communication]. The execution of DSS via web sites raises new and challenging technical and social issues. The use of the internet is influencing changes in both private and working life. So, for example, the use of web-based resources is now commonplace for most managers. Web-based DSS bring the possibility of sharing expensive resources such as large, powerful databases or live on-line data feeds. They may bring effective co-operative working of small groups on the pattern of video-conferencing. But there will also be issues of security, speed and reliability that are of critical concern.

Generic systems versus customised systems

Some of the early pioneers of DSS (Keen and Scott Morton) recognised the failure of computer support to deliver on managers' expectations: *"... the tools available have not been matched to the managers' reality, and the computer remains, at best, of indirect assistance and, more generally, an irrelevant nuisance ..."* [Keen+78]. The same mismatch in technical performance and expectation has been widely noted and commented on during the 1990s. An important part of the problem had already been diagnosed: *"... the*

emphasis in the DSS approach on building systems for specific types of problems, tailored to specific decision situations and decision makers, means that there is no typical design ..." [Keen+78]. That is, the range and variety of business decisions and problems is too great to allow of generic solutions. However, the lessons of history are learned, if ever, only very slowly and rather similar conclusions were again being drawn twenty years later in a project by Adam et al [Adam+98] which concluded that the 'total systems approach' to DSS was unsuitable for organizations. Instead the framework they advocate emphasises the 'novelty and specificity of problems', it is a framework that, *"... is not based on the premise that there are generic types of decisions but rather it allows for the uniqueness of each organization's experience and expertise with the problems that they encountered in specific decision situations ...".*

Thus their approach is organization-oriented rather than problem-oriented. The conclusion to be drawn from this discussion is that in building any DSS a major consideration should be the organization's uniqueness and an effective way to do this is to focus on user-developed DSS. The role of the end-user is crucial in this because although the problems are ill-defined, it is the end-user who, as part of the organization, knows the context and needs (including time-constraints) better than anyone outside the organization, and will be the first to recognise when the problems are identified clearly.

The need for end-user development

In a business environment it is essential that the designers of DSS are aware of, and are exploiting, the latest technology. This makes the development of DSS always challenging and costly. It is important, in order to be cost-effective, to involve the end-users from the very beginning in the process of development, and in ongoing change and modification of DSS to make sure that an organization derives maximum benefit. The major technical challenge here is, can the end-user become a designer?

The lack of any notion of 'typical' design in a DSS led Keen and Morton to suggest the necessity of an environment enabling custom-made design. Finlay proposed something along these lines with his 'Management Intelligence Systems' (MINTS):

> *MINTS are developed in situations where users do not really know where they are going, and thus with objectives not clearly definable at the outset. The approach is an evolutionary one, in which the users can experiment with a small portion of their problem situation and expand their area as they think appropriate (rather than as the designer thinks appropriate) ...*
>
> [Finlay89]

He further comments on the lack of engagement of managers with support systems: *"Managers are happy to let others use systems to provide them with information: by doing so they are failing to utilize the system as a learning aid ..."* [Finlay89, p. 251]. And he cites approvingly the comment of Earl that, *"Managers are seeking answers when they should be seeking insights ..."*. Such comments probably reflect the real situation currently of many managerial staff and their uses of computer support systems. There is an urgent need for business executives to be able to fashion their own formulations of problems, and search for their own solutions, using computer-based support. The reason for this is that business problems and decisions, like personal decision making, are embedded in contexts and cannot be detached without changing their nature. Only the individual managers have the surrounding contextual knowledge and beliefs that, when suitably 'informed', give rise to what Finlay calls 'intelligence':

> *... simply providing [the manager] with information is not enough. Information is the raw material for intelligence, but if it is just given to the manager he is left having to interpret it and include in his scenario without assistance. Where a manager uses resources outside his own capabilities to help in providing a focus, 'intelligence' gathers information and presents an overview of the situation together with a list of options. After questioning and digesting, the manager is then in a better position to take a decision ...*
>
> [Finlay89]

The vision here that distinguishes information as 'raw material' from intelligence that offers 'overview' and requires 'digesting', but also seeks to combine the two, has some similarity with our objective in this thesis of a framework which includes information in the form of observables as well as circumscribed data but combines these with a rich form of human support in the building of cognitive artefacts. This is developed further in Chapters 4 and 5.

Technology for DSS construction

So far DSSs have been built and implemented for a great range of assorted organizational purposes. There are many success stories and, of course, failures. However, as things change and become more sophisticated, DSS will surely be developed further to reach alternative solutions to serve complex modern requirements. It is, however, salutary that in Turban's comprehensive textbook on DSS, published in 1998, he is able to cite as a 'useful framework' for understanding DSS construction the following three levels of DSS technology taken from the work [Sprague+82] written sixteen years earlier:

1. DSS Tools – Software utilities or tools are the elements that facilitate the development of either a specific DSS or a DSS generator. Examples of DSS utilities are general purposes languages, formula languages, visualisation and graphics tools, editors, databases, query systems etc.

2. Specific DSS – The specific DSS refers to a final product or the application that accomplishes a work assignment. It is a special custom made product for a specific purpose in an organization. Most specific DSS are very costly and require a long development time with a team of experts.

3. DSS Generators – The DSS generator is integrated development software that provides a set of capabilities to build a specific DSS quickly, inexpensively, and easily. Different type of generators offer a different package of capabilities such as, modelling, report generation, and graphical display. DSS generator developments may be either:

- Microcomputer-based:- Lotus 1-2-3, Quattro Pro, and Excel are examples of popular microcomputer-based generators which are constructed around spreadsheet technology. or

- Mainframe-based:- Financial Planning System (IFPS Plus) and Express are examples of special-purpose languages for mainframe-based DSS which can be evolved with planning or modelling languages. Nomad2, RAMIS and Focus are examples of special-purpose languages for mainframe-based DSS which are developed around the capabilities of strong `data base management systems'.

The above summary of technology for DSS suggests that current DSS focus on systems functionality and the automatic use of DSSs, or in our terms, they are within the 'engineering' outlook and employ the method-tool-user paradigm (see § 3.4.1). This mentality of automating wherever possible is in contrast to the spirit of DSS as perceived by S. Alter who states that the purpose of business DSS, *"... is to support managers responsible for making and implementing decisions rather than to replace them ..."* [Alter80]. The method-tool-user paradigm when applied to DSS is more suited to operational and perhaps tactical decision making. However, information gained from this kind of support systems alone could not give much support for strategic decisions which involve human experience and capabilities to a much greater extent.

Using general programming languages for DSS building has the disadvantage of requiring users to be knowledgeable in the language or else to be very detailed in their requirements. It also means that subsystems are not usually available and that evaluation is difficult. Formula languages, or spreadsheets, have different kinds of problem. There is no clear separation of logic model and data model, so there is also no clear decision model. The data structures used are limited to two-dimensional tables and the types that are built-in. Some of the comparative features of general purpose languages and formula languages that are used in constructing DSS, based on Nardi's work [Nardi83] are shown in Table 2.

TABLE 2. Comparative features between formula and general programming

No.	Formula Language	General Programming Language
1.	The language allows users to be productive on a small numbers of functions. This is because of its high level and task specific operations. The language allows direct access to the primitives of the language and formulas can be written without having to descend to a lower conceptual level.	The language allows users to be productive on a wide range of functions due to the constructive nature of language that allows the combination of a fairly large number of primitive functions.
2.	The language is less flexible. There are only a certain number of routine functions imposed.	The language is flexible. The limit is the capability of the programmer. This is due to the nature of the language that allows the combination of primitive functions
3.	The learning curve is flat. Once a person learns how to use such a language properly, there is not much one can learn further.	The learning curve is high. This is based on its primitive functions which allows almost endless combinations that one can learn and develop.
4.	There are simple control constructs that provide the power needed; to write conditional statements, to iterate operations, and, to maintain dependencies with little programming effort and with conceptual simplicity.	There are complicated control constructs which are central to accomplish anything. This limits users' progress and creates a serious de-motivating force as the user tries to learn the language.

In addition to the comparative issues in Table 2, Nardi argues that the time-consuming struggle of iteratively and incrementally developing programs is a necessary and irreplacable component of any end-user programming activity:

> *Rather than attempting to provide tools that avoid the need for incrementally working through problems, it will be more fruitful to develop tools that make the struggle as short, attractive, and productive as possible. Spreadsheets have succeeded in doing this to a considerable degree ...*
>
> [Nardi83]

The emphasis of Empirical Modelling on user experimentation and user experience leads naturally to the activity of 'incrementally working through problems' that Nardi is describing here. It is an approach to computer use that accords well with one of the aims of this thesis – to search for an environment that would allow a user-developed, unstructured[1] business DSS. It is also an approach that calls for cognitive engagement on the part of the user and one that gives a central place in the development process to knowledge and learning.

Knowledge-based DSS

The term 'knowledge engineering' was introduced by Feigenbaum partly in response to the perceived inadequacy of formal systems for the analysis of a problem domain. Knowledge engineering refers to the kind of system analysis needed for building expert systems. Feigenbaum and his colleagues regarded it as more of an art than a science with a focus on heuristic knowledge and on contextual skills. A knowledge-based system is built through a knowledge engineering process and is distinguished from other information systems by the inclusion of problem domain characteristics and an architecture in the form of a knowledge base and a reasoning engine. The emphasis in developing knowledge-based DSS therefore shifts from being driven, or dominated, by technical issues to giving priority instead to the problem domain and its context in an organisation. A conceptual model of a problem includes the knowledge and heuristics used by a human expert in the problem solution.

Any methods for the acquisition and modelling of expert knowledge should include both task knowledge and performance knowledge. Task analysis is the first stage of the knowledge elicitation process. It involves the domain vocabulary and theories, and the representation of the subtasks and the flow of control between them. Performance modelling typically makes use of verbalisation by the expert of both description of the human processes involved and prescription of any explicit rules governing human performance. Numerous techniques from cognitive psychology

1. 'unstructured' refers to fuzzy, complex problems for which there are no cut-and-dried solutions [Turban95, p. 8].

such as interview, protocol analysis and repertory grid method (for identifying personal constructs) are used in knowledge acquisition. The task level is expected to identify processes commonly used by any expert in a domain, the performance level will reveal more individual associations and techniques. But the verbalisation approach aims at deriving generalised task knowledge from individual performance knowledge. It should be noted that Klein and Methlie, in common with many other authors, acknowledge that the knowledge involved in this modelling process must be arranged in a particular way so that it can be used in a computer system. More often than not this means a symbolic representation in some propositional form.

Hard versus soft approaches for computing of business activities

M. Pidd [Pidd96] classifies approaches for business computing into hard and soft approaches. These two kinds of approach, according to Pidd, both include methods and tools for systematic thinking and analysis. The hard approach refers to pure quantitative analysis based on machine manipulation of the quantitative data. It includes mathematical and logical modelling (hard management science), optimisation modelling (linear programming) and visual interactive modelling (discrete event computer simulation). The soft approach refers to qualitative analysis based on human interpretative ability alone or with the capability of computer manipulation. This includes interpretive modelling (soft management science), Soft Systems Methodology (SSM), Cognitive Mapping and SODA, and System Dynamics. The soft approach pays great attention to organisational structure and human factors – issues that are neglected in the hard approach. The following is a summary of the main differences between 'hard' and 'soft' approaches after Pidd. See Table 3

TABLE 3. 'Hard' versus 'Soft' approaches after Pidd [Pidd96]

Categories	Hard Approaches	Soft Approaches
Problem Definition	Seen as straightforward, unitary	Seen as problematic, pluralistic
The Organisation	Taken for granted	Has to be negotiated
The Model	A representation of the real world	A way of generating debate and insight about the real world
Outcome	Product or recommendation	Progress through learning

Flynn [Flynn98] discusses the success and failure of the development of 'information systems' and points out that the problems found in 'information systems' are due to the inefficiency of the 'hard' approach in addressing the key problems and underlying assumptions of that approach. These are that problems are logically based and that computer solutions do not need to take account of the social and psychological context. He suggests that the 'soft' approach could help to address the problems raised by the deficiencies of the hard approach. Soft methods such as Soft Systems Methodology, Client-Led Design, Multiview and Participative Systems Design take into account both quality and productivity. To achieve quality requirements, usability and acceptability are major concerns of the soft approaches. Then to achieve productivity, the scope of the organisational environment (both internal and external) and human factors are incorporated into the system development.

Beginning of a shift in computing paradigm: spreadsheet systems

Bob Frankston and Dan Bricklin developed the first spreadsheet program 'VisiCalc' in 1979, as an effective way to use computers to solve business problems. D. J. Power [PowerWeb] reports that a million copies of VisiCalc were sold during its lifetime. The spreadsheet concept is influenced by a traditional blackboard production planning layout. Following 'Visicalc' as the first version of the spreadsheet series, 'Lotus 1-2-3' or 'Excel' are found installed in most modern computer systems.

The major principles underlying the working of a spreadsheet system that suggests the beginning of a shift in computing paradigm are dependency maintenance, combining the logic and data models, and having an interactive visual interface. As well as these technical features the spreadsheet introduced an alternative approach to information processing through the concept of end-user computing.

> *Spreadsheets offer a simple method of obtaining solutions to a variety of problems and are very efficient with regard to the amount of preparation required to obtain these solutions. This efficiency arises from the fact that the spreadsheet already contains routines for data input, data or graphical output and a range of algebraic and logic functions and that there is no requirement for a programmer to rewrite these routines ...*
>
> [Davies95]

The concept of end-user computing is widely accepted as a trend for future use and development. This is owing to the increasing common interest of the general population in using computers to support their daily activities. The majority of people are neither computer scientists nor mathematicians, therefore, the structure of conventional computer programming would discourage them modifying or constructing support systems. A word processor is 'open' in the sense of being easy to update and experiment with text data, while a spreadsheet system is similarly open and easy to update and experiment with numerical data. The view of the spreadsheet as a

'number cruncher' by P. E. Gosling [Gosling89] should be understood in terms of this quality of openness to interaction.

The spreadsheet as a business modelling tool. As the first spreadsheet system was designed to be used as an electronic version of a paper form of spread sheet, therefore, the spreadsheet in general works like analysis papers did for most business people, before the computer era. Although this may have contributed to the popularity of spreadsheets in the business area, there is much more additional benefit, apart from the basic functions of analysis papers, which made the huge investment in setting up computer systems in the workplace worthwhile.

The spreadsheet is appropriate for business modelling because complicated analyses can be performed without using advanced mathematics. In addition, a huge number of calculations can be carried out and the results presented clearly in just a few minutes [Rothery90]. The beauty of the spreadsheet for business practice is its simplicity. Basic business practice has not changed over recent decades. The major differences are in handling the increased amount of data arising from the complexity, rapid change and technological advancement of the modern world. However the complexity of today's business practice does not need complicated mathematical models to handle the issues like those in a scientific discipline. The ideal system for business people might be a system that works like the human brain works in solving our everyday problems but with a more powerful memory capacity and greater speed and accuracy. Business people need a flexible and adaptive system to cope with the complexity and rapid changes in the demands of the modern world.

Spreadsheets offer the basic needs of the modern business world: simplicity, memory capacity, speed and accuracy in handling certain business problems. Spreadsheet modelling is like experimentation, e.g. the changing of cost structure in marketing analysis allows the modeller to use a spreadsheet to experiment with different figures or even with changing the relationship of the parameters (formulae), just as engineers try out models of new car designs in a wind tunnel [Etor86]. Nevertheless, the properties of the spreadsheet are still far from ideal and not all business problems can be

solved by their use. For example, an ill-defined type of problem cannot be worked out by a spreadsheet system. Further discussion of their limitations will follow in the next two sections.

The recent enhancements of built-in functions provide a more managerial toolkit such as a goal seeking function or an optimising function. Scenario Management is another add-in function that allows business people to view different results when different parameters and values are selected for their business model. The macro or short-cut function for repeated work has been expanded with a general programming language such as Visual Basic. With the current versions of spreadsheet systems, the user can include most file types in their working space, e.g. a user can include graphics in their work sheets.

Edwards and Finlay [Edwards+97] have pointed out that the great value in using a spreadsheet as a business modelling tool is the acquisition of 'intelligence' or knowledge while experimenting with the system. They state, *"Intelligence is the product of the meshing together and reconciliation of information. (Note that as used here intelligence has nothing to do with the possession or otherwise of mental ability!) ..."*. Their definition for the term 'information' is, *"Information is data that are or may be useful to a manager in their job ..."*. Whereas 'data' is, *"Data are those stimuli from outside the application systems that pass the rules embedded in the interface between the system and its environment"*.

The spreadsheet allows business people to experiment and to learn from experience. For example, while interacting with their models of 'sales analysis', sales people can find out for themselves how changes in sales figures would affect overall sales targets. The model will enable the optimum price to be predicted in order to achieve the anticipated sales volume required to reach the target earnings, assuming that price is the only consumer concern. With this new knowledge, they can develop their sales strategy. In this case, the sales people have manipulated the available information to provide new knowledge about possible price levels and sales to make a decision on their sales strategy.

The new knowledge in this context is the insight gained through experimenting with available information. It works rather like the way human beings naturally learn in their daily life. For example, experience in cooking would enable a cook to gain insight into the right proportion or the right kind of ingredients required to provide a good dish. This knowledge could then be used to improve an existing recipe, or to create a new one. The creation of intelligence in the context of business activity with the use of a spreadsheet is partly due to ease of experimentation. Experimenting by altering data or information derived from the logic model, data model or both within the spreadsheet, the user can learn how each change affects the outcome. However, because the types of entry and the system structure are limited, spreadsheets allow such experimentation in only certain types of problem.

Edwards and Finlay describe two groups of managerial work that can be supported by spreadsheet systems: planning and control. In production planning, the 'what if' analysis feature of the spreadsheet system can help to avoid human error in dealing with complex data. For example, in a production planning of a flexible packaging company there may be more than twenty different packaging materials, several kinds of machine available, several working processes that a piece of work needs to undergo and special customer priorities (e.g. the top five customers requests should always have a higher priority than the others). This situation is complicated and it could be better to have it managed by a computer to avoid human-error when dealing with too many parameters and data. The bigger the firm, the more complicated the planning or control factors involved. This suggests that a spreadsheet is most likely to apply to decision modelling on well-structured problem types where the problem is clearly identified in terms of mathematical or logical solutions and where the appropriate data are available for manipulating information or knowledge for further decision making. In other words, a spreadsheet is suitable for decision making processes that are routine and repetitive where the nature of the problem is well-structured.

Interestingly, it is not only the work of accounting, financing, marketing or administrating that can enjoy the helpful feature of reducing the work-load of routine and repetitive tasks offered by the spreadsheet model, but also the work of some specific

design problems. Structural engineering design is another possible application for the use of spreadsheet modelling [Davies95]. For example, in an engineering company, the engineering design work to design an elevator consists of solving equations to arrive at providing a proper mixture of the number and kind of materials to meet the specification required by a customer. Such equations, in a spreadsheet system, will be solved by setting up a spreadsheet model that ensures that limiting conditions are obeyed, selecting values from tables or graphs and studying the effects of the use of trial values to come up with possible designs.

Spreadsheet limitations. All the applications of spreadsheet systems mentioned above are, broadly speaking, the application of spreadsheets to problem solving in the spirit of what Winograd and Flores call the 'rationalistic tradition' [Winograd+87]. This is because within the spreadsheet environment, the most recognized application is the construction of a model to analyse data or information in the form of a comparative evaluation of sets of consequences. These sets of consequences are the results of processing either mathematical or logical statements and then listing all the alternative solutions that comply with constraints enforced by the formulae or mathematical and logical expressions.

However, according to Winograd and Flores, the above technique of problem solving by, *"developing a formal model for the system that will be effected, a set of rules that describe the behaviour of the modelled system, and an objective means of assigning evaluations to the resulting effects"* is an idealized situation and can rarely be achieved in reality. They have listed three major factors that inhibit such achievement:

> *1. Rationality requires a complete knowledge and anticipation of the consequences that will follow on each choice. In fact, knowledge of consequences is always fragmentary.*
>
> *2. Since these consequences lie in the future, imagination must supply the lack of experienced feeling in attaching value to them. But values can be only imperfectly anticipated.*

> *3. Rationality requires a choice among all possible alternative behaviours. In actual behaviour, only a very few of all these possible alternatives ever come to mind ...*
>
> [Winograd+87]

This quotation suggests that the use of mathematical and logical reasoning which is found in any formal method of computing is not adequate for all problem solving or decision making processes. To a great extent, spreadsheet systems also work only with either mathematical or logical arguments. However, spreadsheet systems embedded with automated updating, will always maintain an up-to-date state of formulae, parameters and data whenever there is any change. On account of this and in addition to their end-user orientation, spreadsheet systems are mostly used for solving business problems. Nevertheless, the spreadsheet is only well suited to those applications where the problem domain can be clearly identified in terms of a well-structured problem where the representation is in the form of mathematical or logical statements. The quantitative type of problem is a kind of problem or application that is well served by the spreadsheet. Data-oriented problems are another problem type to which the richness in data manipulating facilities (especially automatic calculating and updating) of the spreadsheet applies well.

A spreadsheet package will suit only a limited number of applications. In general, the spreadsheet user is still typically restricted to:

1. only a few types of variable within cells of the sheet: value (arithmetic number), label (text style symbols) and formulae (for defining relationships among individual variables);
2. a certain number of built-in functions;
3. the system's functionality (that serves mainly mathematical and logical arguments).

These inflexibilities of spreadsheets are disadvantages which can be expected from any software package. In addition, a number of spreadsheet users are untrained in

model development using professional system analysis and design and in most cases, they only need the model for their personal use in making, for instance, a rough financial budget or sales forecasting. Spreadsheet models developed in this way lack the three major system principles: reliability, auditability and controllability [Davies95]. A number of research studies referenced in the work of Isakowitz et al have shown that elusive errors are easily made and can cost a firm large amounts of money [Isakowitz+95]. Isakowitz et al proposed a 'model management' tool that could effectively separate the logical aspect of spreadsheet system from its physical representation. This idea was introduced because they attributed a major problem of spreadsheets to the lack of separation of the data model from the logical model.

Another study of spreadsheet systems by Klein & Methlie offered the following reasons why the spreadsheet is suitable only for simple personal applications and not suitable for complex or institutionalized applications:

> *... they do not provide a clear understanding of an application's global logic in using resources such as data, decision models, reports, forms, etc.;*
>
> *... they do not provide a satisfactory readability of decision models;*
>
> *... they do not provide a way to represent easily data structures more complex than two-dimensional tables;*
>
> *... they prevent easy evolution of the application due to the non-separation between data management, models, form and report definition. All these tasks must be accomplished by the user within a grid of cells, a cell being able to contain a piece of data, a formula, and be used for presentation ....*
>
> [Klein+95]

In the real decision making situation, there is not only the range of alternative choices to be considered but also other factors of equal importance, such as qualitative issues (e.g. the effect on customers if the product price is increased). The handling of this type of qualitative issue seems to be done so far only by personal judgement, based on the experience of a particular individual or set of circumstances. Qualitative

issues are more concerned with personal experience, thus a special method of modelling is needed for this area. This suggests that attention should be given to modelling this aspect of decision making. This kind of modelling is addressed by what we have called experience-based modelling.

The spreadsheet is a tool which is suitable only for quantitative or data-oriented models, therefore, those business activities that do not fall within this boundary may be in need of some other sort of tool. Strategic planning, i.e. whether the firm should enter a new market or not, or policy determination i.e. policies concerning new staff benefit, or merger or joint ventures, are examples of managerial activities that need the particular expertise or experience of an individual manager. The way in which the associated decisions are made can naturally be regarded as 'black box' processes within the executive brain. If an alternative system could be developed that could provide explicitly simplified versions of such complicated models of perceptions, it would offer a much richer support system for decision making.

Systems for data-oriented models have been successfully developed and have been the subject of much research. This might be because the underlying structure is based on mathematical models which have been widely accepted through their successful use in many scientific models. The social science features of business practice are indeed in need of an alternative approach in order to reflect the real impact of qualitative issues in the models being developed.

Management science view of DSS

Company decisions need to be made quickly and continuously. Both internal and external updating of information and data are important. Computer applications are shifting the focus of decision making from transaction processing and monitoring activities to problem analysis and solution applications. Data access, on-line analytical processing, the use of the internet and intranet are areas of much interest for the coming century in the field of management decision support in making business decisions. The key benefits of using computer systems are: to increase speed, to reduce cost, to improve quality and to overcome cognitive limitation. Performing complex

simulations, checking many possible scenarios and accessing a range of outcomes at the highest speed and the least cost possible are motivation for the use of computer systems.

If they are following a management science approach, managers perform their problem solving processes systematically. Thus a scientific approach can be applied in order to partly automate managerial decision making. A typical systematic process of problem solving might be the following:

i) defining the problem;

ii) classifying the problem into categories;

iii) constructing a mathematical model to represent the problem;

iv) finding and evaluating the potential solutions and

v) choosing and recommending a solution .

The explicit data, information and knowledge are kept in the computer and will be manipulated when there is a need. Decisions are often made on the basis of partial, incomplete or inexact data, information or knowledge. Furthermore decision makers recall experiences and learn from their experiences what to do with new situations for which there are no exact replicas available. In a DSS using Artificial Neural Networks (ANN) which uses a pattern recognition approach, there is the possibility of dealing with incomplete situations. Other DSS technologies offer little or no element of machine learning. Within ANN, the greatest advantage claimed is the capability of neural computing to continue working despite missing data.

A key characteristic of decision support systems, according to [Turban+98], is that they should include at least one model. Then the analysis of the DSS is based on the model(s). There are three classes of models, with increasing degree of abstraction, that are used to represent systems or problems. They are:

i) iconic – all kinds of physical replicas of systems;

ii) analog – this includes symbolic representations and some physical ones, they do not look the same as the real systems, but they behave like it;

iii) mathematical or quantitative models.

Only recently has it been possible for advanced technologies in computer graphics to allow the use of iconic and analog models, on a commercial scale, to complement mathematical models in constructing management support systems. An example might be a visual simulation where the modelling involves the conceptualisation of a problem and its abstraction to quantitative or qualitative forms.

2.4.3 Future Trends

Gray and Watson have identified a new direction for DSS that has certainly not yet exhausted its potential: "*The new DSS is the result of two software solutions needing and finding one another: Data base firms developed data warehouses and were looking for applications. EIS and DSS software developers and vendors needed to deal with ever increasing data bases ...*" [Gray94]. It seems likely that the latest development of this kind of data base architecture, with ideas from data warehousing and data mining will help to deal with data rich environments in the next era.

Sauter's observations that, "*Over the years ... expert systems have evolved into an integrated component of many decision support systems provided to support decision makers, not to replace them ...*" and "*... expert systems (or intelligence) technology became a modeling support function, albeit an important one, for decision support systems ...*" [Sauter97] seem to point out that considering the use of expert systems as supporting tools for DSS cannot be neglected.

Another technology from AI that might be used in modelling for DSS is that of multi-agent systems in which collections of intelligent software agents cooperate to

achieve a common goal. Such methods have recently been advocated for simulation modelling [Gilbert+99].

In conclusion, it is likely that the future trend of DSS will build on environments that use newly developed database techniques and architecture and that allow building architecture for expert or intelligence systems that will be well represented by means of simulation.

2.5 Issues of business support systems

Since the first development of computer-based support systems and their use as administrative tools for accounting, by people in business, there has been a tremendous change in their speed and power due to the dramatic changes in technology. But there has not been a corresponding development in principles for software or tools. The current principles are implicitly burdened with practical difficulties as follows.

1. **Separation between users' experience of the world and their experience of computer support** – The existing uses of business computer-based support systems still apply methods in computing where there is a separation between the user's experience of the real-world and the user's experience of computer-based support systems. This is because of the strict structure imposed by program conventions. That is, to maximize the computing capacity of the machine, instructions, inputs and outputs are in the form that best fit the structure and the processes of the digital machine. These may not necessarily fit well with the understanding of human beings. For example, programming conventions require the users to be able to interact with certain data types or data formats which may not match well with what is experienced by the users when interacting with the real-world domain.

2. **Passive and limited interaction limit the opportunity to develop new knowledge** – In the early days of the computer information systems were referred to as 'data processing systems'. Regardless of the name applied to them, they are systems built for collecting and manipulating data. This type of support system is generally a passive support system in the sense that even when it can help in analysing the domain, its structure does not allow the

incorporation of what has been learnt by the users during the operation of such models. In this respect, it limits the evoking of new knowledge that often arises when we incorporate new experience into our so-called 'knowledge-base'. However, there are some subsystems under the general term IS such as MIS, EIS, DSS and ES that, in some ways, work actively in response to users interactions. For example, an inference engine of an expert system works in correspondence with users' enquiries and criteria provided. Nevertheless, there is limited interaction that can be made between the human agents and the computer-based systems. Most importantly, when there is the possibility of interaction, it usually has a fixed interpretation determined by the designer. Such interpretations are shaped dominantly by the designers' background or how they perceive the domain subject, which may not be the way users perceive it. A system designer may not be aware of the significance of choices of interaction that can affect a particular user's cognitive process. That is, for example, in interacting with other people, or the world in general, the openness and flexibility of individuals' interpretation and interaction is often vital to their learning. Because of such properties – being passive, having limited interaction and narrow scope for interpretation – such systems do not allow human agents to learn or develop much new knowledge which often occurs through *interactive experiencing* and *experimenting* with the object of interest.

For example, without having a real experience of driving a car yourself, you will never know how to cope with the situation at hand. However, there are a number of conventional systems that are interactive. For instance, the simulation for pilot training is a kind of interactive computer system that enables the development of new knowledge but with a very limited scope. That is a pilot trainee can learn and gain new knowledge of how to fly an aircraft. But such new knowledge is bounded by the context set by such a system. The context is set by the range and number of variables used, the nature of interface and controls, and the detail with which the environment is modelled. That is why after the mock up training (flight simulation), a pilot trainee still needs an extensive actual flight training before flying any commercial aircraft.

A similar interactive system example is the 'Expert System'. For example, a trainee doctor may learn new knowledge from a Disease Diagnosis Expert System. However, what can be learnt is derived from what has been put in and the scope of the interaction allowed. A trainee doctor may gain new knowledge of the possible symptoms for a disease. However, it is difficult to deduce any association between the recently developed symptom which has not been diagnosed and put in the system yet with what has been observed by a doctor who wants to find out more about it. In spite of the fact that such systems enable interactive experience, such interactive experience lacks the possibility of open-ended experimenting. Here 'open-ended' means going beyond the preconceived context in the above sense, developing new contexts, and acting opportunistically. This is because, in such a kind of system, the interpretation of any interaction is pre-determined. Open-ended experiment, which is rooted in open-ended interaction, is a unique and important source of learning where new knowledge is developed. The difficulty of open-ended experimenting is part of the nature of conventional computer-based support systems. This is owing to the emphasis on automatic operation in existing computing paradigms.

3. **Assuming universal use and neglecting the need for adaptability to cultural or individual characteristics.** – Social factors are undoubtedly a great influence on business practice, which is a social system. For example, norms, attitude, culture, traditional, belief, perception, practice and value, are always different in different social systems. These factors shape the understanding or interpretation of humans towards particular thing(s) or situation(s) [Goffman61, pp. 40-42].

An example of the significance of difference in cultures is as follows. Consumer rights are not so well protected in Thailand as compared to England, thus a marketing strategy which applies in England might not work the same way in the Thai market. This kind of thing, if it is not well understood, could lead to constructing either a poor strategy plan for the company or a poor support system. This sort of fact has not been taken into account in most of the

current approaches that are used to develop computer-based support systems for business activities. This is especially relevant to those countries that are not leaders in computing developments. For example, most of the leading business firms in Thailand have to employ computerised support systems that were originally developed for their headquarters abroad. Staff are concerned that they must familiarise themselves with the structure of systems that were not designed for them.

In general, most of the manual systems with their individual adaptations which firms employed, lost their individual character when they were transformed into computerised systems and they all had the same structures and functions. The constraints imposed by conventional ways of computing, in which the computer-based system is treated as a 'mathematical machine' [Chaitin99] or an 'information appliance' [Norman83], did not allow for the preservation of individual peculiarities. These conventional ways of thinking concentrate on the power of the computer e.g. emphasising its speed and accuracy. However, to maximise the full capacity of the machine at the expense of losing responsiveness to the individual might not be the best way to use a computer-based system .

Business problems and needs in general

Applegate and colleagues [Applegate+96] suggest that business needs and problems in relation to computer-based application systems arise from re-engineering within a business. That is, changes in society and world economics have led to transformations within businesses to information oriented activities. They point out that only recently has the cumulative impact of the technological, social, and economic adaptations forced business practices into a radical change toward a new business model, despite the evolution of information technology over the last thirty to forty years. They state, *"Information is now a critical ingredient in production and a prerequisite to the development, coordination, control, and allocation of other resources ..."* and *"As the economic value of information increases, the economic rewards accruing to those who can access it, transform it, and use it also increase ..."*. This supports the need to pay attention to the necessity of

fundamental improvement in the core business success factor of today: effective information systems.

Computer-based systems have proved their success in structured business problems e.g. routine or repetitive problems. The practice of accounting and financial planning in most firms is conducted by either specific developed systems or commercial system generators. However, most firms are still looking for powerful supportive tools for their decision making processes for those unstructured business problems such as arise in making strategic managerial decisions. For the past few decades, there have been a number of systems which were expected to have potential for these purposes. For example, MIS, EIS and DSS are the best known tools, but they remain limited. One of the most difficult features is to give cognitive support to users. That is, to support a variety of kinds of knowledge such as propositional, experiential and tacit, and to support a range of qualities in knowledge and belief such as vagueness, uncertainty, subjective, provisional etc. Recent research in AI in topics like machine learning and fuzzy theory may contribute to this difficult area and provide a valuable additional capability to information systems.

Business problems and needs in particular: decision support

There are some features that are common to all decisions. According to one author on DSS, *"In any decision there are three elements, all different in nature but which apply in problems large or small, public or personal. These are: 1. the range of choice; 2. the consequences of each of these choices and 3. the objective(s) involved ..."* [Rivett94].

Daily business activities deal with numerous decisions, but not all business decisions need the support supplied through the construction of models. There are many situations where the decision can be made simply because none of the three elements stated above is critical. The decisions that are considered to be critical and may need a business decision model, according to Rivett [Rivett94], are those decisions that possess one of the following properties: i) there is no easily available, acceptable and valid unit of measurement for relevant parameters; ii) the range of choice of courses of

action is uncertain, or, if known, too large to be able to consider each of them; iii) the consequences of these choices are uncertain and iv) there is more than one objective or even, perhaps, no agreed objective(s).

For example, suppose a marketing department has to make a decision whether to enter a new foreign market or not. This kind of decision needs more than simply the population of each prospective market to make a decision to enter such market. In this situation, the unit of measurement on how attractive each market is, needs a combination of a number of units of measurement e.g. size of population, standard of living, purchase index, economic growth, the political situation and government stability. It can be seen clearly that some of these units of measurement e.g. the political situation and government stability are difficult to measure, validate or gain acceptance. There are also so many alternative markets to enter and the consequences of selecting any of them as a target market are quite unknown because of the variety of factors involved e.g. consumer behaviour, culture and buying pattern. Finally, the overall aim or objective of the firm, in theory, is the force that shapes the final decision. But it is common to find that there are many hidden objectives (mostly personal objectives) that influence the shaping of the firm's objective(s), or cause a dispute among management members. This highly influential decision element makes the business decision even harder to manage. Then there is a need to develop business model(s), for example, a financial analysis model that would enable business people to have a good picture of what would be the overall benefit for the company e.g. the estimated return on investment on entering a particular market. The model(s) would then provide business people with a tool that they could manipulate to gain more supportive information for further decision making. However, it can be seen clearly that there are still a number of factors that are qualitative in nature and deserve equal attention. So far, these qualitative factors which are a crucial aspect of decision making have not been effectively included in computer-based business models.

Personal experience in different types of business, both as an observer of and participator in the decision making process, suggests the following situation. Firstly, decision makers rarely have a chance to assess all the supportive information neces-

sary prior to making decisions. Often therefore decisions are made based on personal judgement or intuition which may be regarded as an intelligent judgement of a particular person. This will have its limits. In the theoretical business sense, it may be considered highly risky to base a business decision on trust in a personal judgement or intuition. That is why a need to implement more scientific methods to assist in the decision making process has emerged in a variety of contexts.

Secondly, information or data that has been handed on as a supporting tool is often, if not in most cases, presented inconsistently, having regard mainly to the original expectation or idea of the person from whom it originated. Such inconsistency or bias could be avoided if there was a means of coordinating such information or data. In general, this situation is a result of different viewpoints on the same thing between a person who made a request and a person who responds. Imagine how much easier it might be if the one who requires the information is also the one who provides it.

In the pre-computer-based era, the calculator was a useful piece of good office equipment which provided support to decision makers by helping them to derive more accurate figures quickly. From the early period of the computer-based business process era (1960s) to the information age of today (2000s), computer-based systems have developed as office equipment that will supply accurate figures and facts (text descriptions) in even less time than the use of the calculator. In addition, there are a number of evolutionary features that are provided by computer-based systems such as analysing, modelling and simulating all the possible scenarios for very complex business problems.

Decision Support Systems were introduced in 1971 by Gorry and Morton [Cortada96, Turban95]. However, the long established record of something does not necessarily mean that it has been well founded. The development of DSS has been subject to many obstacles. The most outstanding are: the ***dynamic nature of business practice*** and the ***restricted nature of the current practice in software systems development***. There is still room for both academics and practitioners to search further for novel approaches to support business needs.

Trends in business needs for support systems

The trend for business practice in the 21st century – transforming raw data into useful information or knowledge and delivering it to a particular unit of operation that can make the most use of it – has been given a high profile in much contemporary literature on information technology. Dhar and Stein [Dhar+97] introduce a 'Knowledge-Intensive' approach to problem solving. Czerniawska and Potter [Czerniawska+98] introduce the concept that information is a more valuable business entity than tangible ones. For example, information about potential buyers and their needs may be worth much more than the current stock of a certain product. How much value to put on such information is controversial, but beyond the scope of this thesis.

In such a way, a thorough analysis of a huge range of available data may reveal a substantial hidden opportunity for either reducing costs or increasing profit. For example, the analysis and manipulation of a customer database could yield information that a particular buying pattern depends, or does not depend, on the nature of a company's offer. However, this practice of data analysis needs both high quality computer-based systems and high quality human systems (expertise of people) [Dhar+97, Czerniawska+98]. The concept of data analysis of this kind, is known as 'data mining', that is, using an application system to search for data that corresponds to a set of criteria out of very large data sets. An example of this usage is a use of an insurance company to provide information for their customers on benefits from different schemes based on individual qualifications and conditions. Data mining is also a technique used in expert systems or knowledge-based systems.

The new era of business practice, with its emphasis on 'knowledge' as an asset of prime value, needs an appropriate combination of computer-based systems and human faculties in any specific problem solving situation. This is mainly due to the following matters of fact:

1. computer-based systems are generally good at handling huge volumes of data with speed and accuracy but lack creative thinking (e.g. how the data should be manipulated without an instruction from human), while humans are

generally good at creative thinking (e.g. what are the core factors that should be involved when making a certain decision);

2. humans are capable of learning from experience (e.g. an experienced manager would probably find it naturally easy to make a judgement after investigation, about the advisability of entering a target market by referring to his experience of similar kinds of market situations), but lack speed and accuracy when handling a huge volume of data (e.g. without the use of computer model in preparing an accounting report, resources of staff and time will be wasted in the time consuming process of manually calculating and recalculating a huge volume of transactions in order to derive certain pieces of information);

3. humans are good at common sense decision-making. This ability cannot yet be provided to any significant degree by any computer-based system. This qualification leads companies to abandon or shorten unnecessary processes or procedures to be run by the computer-based system, if such a task is performed in integration with humans.

In addition, because of advances in modern computer technologies (e.g. a revolution in computer processing systems from inflexible mainframe operation to flexible distributed workstation operation), and the high sensitivity of information changes on business results, real-time or interactive computer-based systems are found to be the common systems in use. For example, the stock markets anywhere in the world use real-time interactive systems, as a single entry either to buy or to sell at any moment of time could easily make the company or the investor lose or gain a considerable amount of money. Thus in this particular situation, the stockbroker also has to have interactive real-time access to financial models to be able to get supportive information while making an investment decision.

Dhar and Stein classify the uses of computer-based systems in business according into two categories: transaction processing (i.e. accounting records, production records, customer records, human resource records and logistic records) and problem solving (i.e. financial investment analysis, corporate resources planning and marketing analysis and planning). Record keeping or transaction processing operations are gradually improving in line with technological developments but there has been no

very significant step change. On the other hand, problem solving operations have changed radically in recent decades because of the improvement of communication networks, the latest technology in database management and the widespread use of workstations or PCs.

2.6 The Need for an Alternative Environment for Business Activities

The evidence of this chapter shows that most computer-based support systems, even those which claim to support executive management decisions are, in practice, able to support decision makers only with 'detached' idealised scenarios. That is, the decision makers are only supported with abstraction-based facts and figures or, in other words, public knowledge. This kind of public knowledge is useful in terms of advocating general ideas. However, it does not help either to discover real issues in a situation or to generate new ideas which are essential factors for making decisions. Abstraction-based facts and figures are the only forms of data and information that can be observed from contemporary decision support systems. In addition, the interaction allowed in such systems is simply feedback in the same context. These forms of data, information and interactions are more suited to describing a predictable phenomenon than the volatile nature of human activity.

Dealing with volatile environment as people usually do with their everyday problem solving, human experiences are a basic natural mechanism for how people understand things and develop their knowledge of how to deal with the world. A business decision support system which is a tool to support business decision makers in performing their tasks better needs to correspond well with how people utilise their knowledge to deal with a situation or a problem. To achieve this aim the system needs to manage not only rich data sources but also a range of interactive models of the problem so that the user can become part of a genuine learning process. They must integrate principles and concepts from a variety of disciplines: cognitive science, psychology, management science and computer science. It is widely recognised that

knowledge is an important business asset but so far there are few support tools that are capable of representing knowledge in a form useful for managers.

The results of the review also shows that the tools currently most used in constructing DSS, such as formula languages and general programming languages, still have shortcomings stemming from:

i) A trade-off between the functionality and the flexibility of software systems. That is, the more functionality a system has, the less flexibility we can expect of a system. The trade-off of this kind is rather crucial to a business use of computer systems. This is because a rapid change is constant and there are few theories of business practice.

ii) A mismatch between users' needs and what software developers can provide. This might be the developers misunderstanding what the users want or the users not knowing what the computer systems can provide.

iii) The difficulties of abstraction-based methods employed in current computing paradigm that may prevent novice users being involved with the actual activity of software systems development. In fact, they need to be involved very much with such activity because it is aimed to support their tasks.

The result of the analysis above leads to a study of the Empirical Modelling (EM) approach and its tools because they offer the prospect of an alternative environment for user-developed DSS. This is because Empirical Modelling has a philosophical foundation that gives priority to human experience. EM puts a great emphasis on providing an environment within which those who use computer-based systems to support their task activities can develop and adapt the systems in a natural fashion.

Chapter 3

Empirical Modelling as a Paradigm Shift in Computing for the Business Environment

3.1 Introduction

In the previous chapter, the approaches, tools and technologies of computer-based support systems for business activities have been reviewed. The difficulties and current issues in the contemporary use of computer-based systems to support business activities have also been discussed. The mismatch between the needs of computing for business and what can be supplied by current computing approaches has been identified. This situation motivates a search for other possible ways to exploit computer-based systems to support human activities which are rooted in the business world. The use of mathematical and logical methods in programming, while they have proved successful in the field of experimental sciences, are not necessarily well suited to the social sciences, and to the business area in particular. How spreadsheets have alleviated some of the difficulties and are the beginning of a new paradigm in computing has been discussed already. The limitations of spreadsheet systems have been analysed and this prompts the idea of an environment preserving some principles of spreadsheets but overcoming their limitations. Such an alternative environment for computing, called Empirical Modelling, corresponds better with the needs of the business environment is proposed.

In this chapter, fundamental theoretical thinking about why EM is appropriate to the business environment are put forward for the first time. The chapter explains why the current ways of constructing computer-based support systems using computing conventions do not fit well with the needs of business people. The final section points

out how the properties of both spreadsheets and EM models, when viewed as cognitive artefacts, make them well suited for supporting human tasks, especially problem solving and decision making.

3.2 An Empirical Modelling Approach

The Empirical Modelling project includes both philosophical thinking and the technical development of modelling notations. Much of the past technical research work has focused on interactive systems of many kinds. Some applications have also been made in design and graphics. However, little of the previous work has been devoted to the application of the approach to the field of social science and business practice in particular. This research work is one of the first substantial attempts to focus on issues of how the approach can be applied to the area of business processes so as to explore its potential benefits and limitations.

3.2.1 The principles

Two central concepts: observation and experimentation have led EM work to develop and expand in a different direction from the conventional approach. The focus in conventional computing is on *optimised automated activity,* with less attention being paid to the consequent trade-off, the *loss of original properties of the domain* and *loss of human users' involvement.* This loss occurs because of the way in which machine capacity is utilised: it is assumed that the task needs to be performed automatically. To achieve this most of the domain properties are modelled by formal statements of mathematics and logic based on programming principles, so that all of them can be further transformed into machine code to be automatically operated with at the machine level. Such transformation processes cause a separation from the world of experience and make revision very difficult without redesign. The EM focus is on how to utilise the machine capacity while paying attention to how more properties of the domain can be included as required. This requires a reconsideration of state which is discussed in more detail in § 3.3.2.

Emphasising the role of experience leads to the need to pay attention to the conditions required to have experience. The first is the *human agent* who is the only agent in the system that is capable of having experience and describing the experience. The second is the *context* (situatedness of the circumstance) in which the human agent has experience. The last component is the *interaction.* This refers to the interaction between the human agent and the context which defines the reality of experience. The interaction referred to here ranges from a subjective kind of interaction to an objective kind of interaction. An example of a subjective kind of interaction is a human agent's awareness of a particular element of the environment. Such awareness does not exist if the human agent does not pay attention to the element. An objective kind of interaction is more obvious and often found as a usual kind of interaction referred by conventional computing. An example of objective interaction could be as simple as typing in this sentence, or a more complicated one as entering a long command to process a range of inputs from different sources.

People can concentrate best when they engage with the object of interest. This leads to the idea of constructing a computer-based system through the use of computers and software as 'cognitive artefacts' instead of merely automated machines (detailed discussion proceeds in § 3.3.3). That is, to have a computer-based system to work in a way that will support cognition, in addition to performing automatic tasks, e.g. calculation, searching or sorting. This is supported by the unusual nature of an EM environment (see Chapter 5) with its distinctive mode of knowledge representation (see Chapter 4) together with its flexible and open-ended interaction environment. Such properties enable an EM based model to serve for a modeller, in a similar way to how a model of a building made of cardboard serves for an architect, that is, to support cognition.

A computer-based system viewed as a *cognitive artefact* offers a distinctive quality of interaction compared with one viewed as a virtual machine composed of *abstract objects.* The concept of cognitive artefact refers to a specific instance which may be represented pictorially in an EM model. The concept of *abstract object* expressed in mathematical and logical statements in a conventional program does not refer to a

specific instance. So cognitive artefacts in an EM environment offer an experience-based kind of interaction. Experience of the cognitive artefact is representative of experience of the real world. For example, a pictorial representation of the model offers a kind of direct association of having interaction with each component or the whole model, in the way a car designer interacts with parts, or the whole body of her car model. That is, she can interact with her car model as if she were having actual interaction or experience with it. Nevertheless, she has to represent her experience in the real-world with another experience with the car model.

Cognitive artefacts provide the user with a representation of the current status of their referent. This means, in the case of a problem solving process, that such a cognitive artefact would actually support the process in an interactive way and may lead to knowledge of certain facts about the situation. This is based on the fact that, to most people, a *cognitive artefact*, which is often represented by a visualisation in EM for the system or situation being studied, enables physical contact (e.g. seeing or touching) and makes more sense than purely abstract mathematical and logical notations. An artefact model as a representation of a system or situation is a more common medium of human experience than a conventional program.

The properties of being *experiential, situated* and *human-centred* are major concerns when describing the state of a cognitive artefact in an EM approach as opposed to the properties: *circumscribed, pre-defined* and *mechanical* which describe the state of an abstract object as used by a conventional approach. The property of being *experiential* refers to the observable, experimental and explanatory quality of state. 'Observable' means any entity that is derived from observing things as they happen in everyday life. 'Experimental' means an act during the development process when a test is made to examine the validity of a hypothesis. 'Explanatory' means that the system represents some reasons, or causes, for the states and state-changes that occur. The property of being *situated* then refers to a consideration of state as experienced here and now rather than in an abstracted sense. The term 'state' is employed to cover three main meanings within a particular context. These are:

- state of the computer system representation;
- state of the human agent mental model and
- state of the real-world situation.

EM gives equal significance to both the computer and the context. The final principle of EM, which is the *human-centred* element, takes seriously the significant issues surrounding the human capability to learn and develop a skilful action.

The way in which a computer-based system is built up in the EM approach depends on construing (i.e. forming a personal analysis, or understanding) a phenomenon in terms of three basic concepts: observable, dependency and agency in the following way. First of all, the entities present within the system are the observable elements (the elements of the phenomena being observed or studied). These entities can be either independent or dependent entities. If they are dependent in the sense of an indivisible dependency such as expressed by the rules of a game or physical constraints, their interrelationships with the others have to be described in terms of dependency by tools called 'definitive notations'. If they are independent, their descriptions are described as individual properties and need not to be based on any other entity. After that such observables are grouped into 'agents'. The agency structure is used to classify the observables of the phenomenon into groups. Each group consists of an 'agent' which is the name given to the group of observables as a whole. The agent is able to initiate the state change of its elements and to effect a state change in the phenomenon as a whole.

The three basic concepts mentioned above cannot be explicitly represented by a hierarchical structure. Instead they are fundamental aspects of experience which people use to make sense of the world. Current situations can be helped by past experience, not only in a textual way but also by the images of a situation which that experience generates. A relevant chain of events need not occur in sequential order, or in any arrangement of hierarchical structure, but only needs to bear some relationship to the situation at hand. These structures are based upon the modeller's observation

and interpretation of the domain or, in other words, the modeller's experience of the domain. The general principle of development for a model or artefact in EM is the correlation of one experience (of the domain) with another experience (of the artefact). Whenever an observation or behaviour in the model does not correspond with the domain then new observations or redefinitions are introduced into the model. In order to establish the reliability of a component of the model experiments are conducted on the component until there is sufficient coherence between the behaviour of the component and the behaviour of its real world referent.

In general, an EM model is a representation of a state of the domain based on the interpretation of the system builder or domain expert who is one of the many possible agents of the system that could introduce a change of the state. Usually, the modeller is not 'inside' the system but can act outside the system as a kind of 'super-agent'. Another agent, for example, could be the system clock, the customer, the member of staff and the world-bank interest rate structure. An agent 'system clock' introduces a change of the state by the changing of time that initiates a certain action in a real-time operation. Agents *theCustomer* and *theMemberOfStaff* introduce different kinds of state changes as the reaction of individuals to a company policy or an external event. An agent 'theWorldBankInterestRate' introduces a state change by the effect of its changing rate that will implicitly alter an individual local bank interest rate. That is, usually a local bank interest rate very much depends on the World Bank's policy. To reflect such dynamic change of state the EM approach is applied rather than a conventional one. This is because by using an EM model, flexible predictions can be made which will enable a more immediate response. Unlike conventional programming, EM, at any rate in the preliminary stage of model-building, gives only secondary consideration to control structure, and gives primary consideration to state. Automatic updating of dependency which is the implicit action of the computer system, or the explicit action of an agent are the only sources of state change. The loosely controlled structures, i.e. what is going to be input, output or the operations, make EM, in a programming sense, more primitive than a conventional approach. This property makes an EM computer-based system very flexible.

With a traditional approach, the modeller has to plan the inputs, outputs, and operations necessary before starting the construction of the computer-based system in order to prescribe the processes involved. These processes are, generally, structured sequentially according to the series of actions that are going to take place. If there is a new input, output or operation, for example, an additional system feature is required, the whole process may have to be revised and redesigned. This is normally expensive. With the use of an EM approach for system construction, any new feature of the system can be added in at any point during the modelling process through redefinitions without the need to revise the whole process from the very beginning as required in traditional programming. This is supported by the specially designed tool called *tkeden* which is a general system interpreter for definitive scripts.

In describing the state of an artefact as experiential, situated and human-centred, a crucial element is the open-ended interaction that is performed by the human subject. Referring back to the earlier discussion, the ease of adding in a new variable (or an observable in EM terms) in comparison with conventional programming, offers a distinctive quality to human-computer interaction. This is because it enables the flexibility needed to make rapid changes, and an immediate response to any new observation made, which may lead to new insight. With the new observation added in, or with the changes in the processes during the interaction of the modeller with the computer-based system, it is likely that other possible parameters, or new observables, will arise in the modeller's mind, apart from those initially expected. This is one of the crucial characteristics of human faculties that cannot normally be imitated by any computer-based systems. Observations related to experience develop new understanding of new ideas.

3.2.2 The EM environment

EM can best be regarded as an approach for creating environments rather than a methodology for developing systems. This is because EM seeks to establish the reliable components in a domain from which a system might be developed. There is no general methodology for this process. A simple example of the processes of constructing

a model with definitive scripts is presented in Appendix A. Any model contains a script of definitions that represents the observations and dependencies regarded as relevant in a domain together, possibly, with actions and functions. The following are the components of the EM environment.

EM's LSD account. The initial account of the three concepts of an EM computer-based system, namely *observable, dependency* and *agency,* are organised and expressed as an LSD account of a domain. (LSD is a name, not an acronym.) An LSD account is much more flexible than the conventional system specification and at the same time it is where the initial system analysis takes place. That is to say, an LSD account is a way of documenting emerging patterns of interaction. It does not provide a context for future interaction, but does provide a focus for potential experiment subsequently. This use of an LSD account can be usefully compared with the use of notebooks by M. Faraday to record the experimental process that could be followed through by other scientists [Gooding90, Ch. 6, 7] and [Gooding01]. An LSD account is flexible enough to change so as to reflect the changes in the ideas of the modeller, in other words, to record the modeller's thinking. That is, an LSD account is used alongside the model building.

The major distinction between an LSD account and a conventional system specification is best described by the following analogy. The LSD account is like the sketch of an artist just as the traditional system specification is like the blueprint of an architect. The purpose of the artist's sketch is to introduce a soft boundary to precede the detailed artwork whereas the blueprint of the architect is a hard boundary that enforces strict attention. Certainly, the hard boundary of an architect's work can be altered but only with extra cost (the whole structure has to be recalculated and redrawn) while the changes in the soft boundary in an artist's sketch could bring a new look to that particular piece of work without an additional expense. However, there is still even some danger with having a soft boundary. That is, if an artist has altered her painting in a new direction, then, it is nearly impossible to recover her previous vision, unless she restarts her work on a blank canvas. Nevertheless, the point

to make here is that, a final painting may fit very closely with the artist's vision and similarly a computer- based support system may fit closely with the user's needs.

The major components of an LSD account are the agents and their subordinate elements – observables in EM terms, which are named according to their properties. The possible kinds of an agent's observables are classified into the following categories: state, oracle, handle and derivate. These agent's observables are analogous to variables in a conventional programming sense. 'State' represents an observable that shows the current status of an agent to whom it belongs. 'Oracle' represents observables to which such an agent can respond. 'Handle' represents observables to which an agent can make changes. 'Derivate' represents observables that are dependent on other observables. The last part of an agents description is 'Protocol' which represents the privileges of an agent to make redefinitions. The following is an extract from an LSD account for the model of traffic lights (Source: Notes for MSc module on Empirical Modelling for Concurrent Systems, written by M. Beynon).

```
   agent      switch_in_road

      {

   oracle     car_crosses_switch, time

   state      car_waiting, time_to_clear_lights,
time_crosses_switch

   derivate   elapsed_time = time - time_crosses_switch

   protocol    car_crosses_switch    ->    car_waiting=true;
time_crosses_switch=|time|,elapsed_time>
time_to_clear_lights -> car_waiting = false

        }
```

In this account the expression |time| denotes the time as evaluated in the current context. An LSD account is established as a rough sketch of the system. It is not a prerequisite to the construction of an EM model. An expert EM modeller may directly implement the construction of the system without an LSD account. This is in line with the analogy made earlier about the artist and her sketch. An experienced artist may have a clear image in mind of her work and can simply paint it straight away without

any sketch. The novice artist may find the practice of sketching helpful in organising her ideas for the painting. Likewise, a novice EM modeller may find a structuring of an LSD account useful to organise and structure her thoughts on the domain under study, while an EM expert may have in mind a clear outline of an LSD account. In addition, as a sketch may be of use at any point in the construction of an artwork to gain a different view of the artwork in its composition, an LSD account could likewise be introduced at any stage of the model building to share or have a different perspective on the model.

Definitions, functions and actions. Whether or not the model has an LSD account, the modeller can proceed to build a computer-based system by EM approach. An EM model is the representation of 'state' by definitive scripts that use definitive notations as media [cf. Beynon97a] (see Appendix A for more information and an example). Beynon [Beynon97a] stated that, *"... a definitive notation is a simple formal language in which definitive scripts can be formulated ..."*. Though Beynon was the first to use the term 'definitive notation', the core concepts have been used elsewhere. For example, they are in the work of Todd on the relational algebra query language ISBL, and of Wyvill on an interactive graphics language where a definitive notation is first used in interactive graphics. Beynon [Beynon97b] indicates that both definitive scripts and LSD accounts are of the linguistic construct type of programming that is intended to describe external observables in accord with how such phenomena make sense to the modeller. This is unlike a program written in a high-level language that consists of variables and specific values, but more like a spreadsheet formula language which consists of values and relationship variables to-be-determined in the context of their external environment.

EM models make use of a definition-based system of which a spreadsheet system is an example. Definition-based systems work on the basis of giving definitions to observables whenever possible. Such definitions express dependencies which the system automatically maintains if changes occur. The definitions may be 'nested' arbitrarily deeply, the only constraint being that no 'cyclic' dependency may be introduced. By this means, the definitions are always kept up to date. These proper-

ties are essential to help avoid error in a complex system or a system that is often changing. In addition, the expense of system maintenance of this kind of system is low compared with the conventional procedural system. This is because any necessary alteration of a system feature can be done on the particular definitions concerned. In contrast in procedural programming, large parts of the codes may have to be revised when any change are introduced.

EM shares the basic dependency principle which underlies a spreadsheet system. Such a principle provides for the automatic updating of the value governed by a formula which follows from dependency maintenance. EM employs a special feature in its underlying programming – the definition – in order to express such a dependency. Dependency-maintenance is automatically performed by the computer system without user involvement. Dependency is an implicit operation that is activated automatically by the computer system. Such an implicit operation is to re-evaluate the definition whenever there is a change in the definition. That is to say, making variables consistent with their definitions. For example, a change of value of *'a'* from 4 to 7 from a redefinition done by a human agent will lead to an automatic trigger of redefinition by an internal environment – the computer system (*tkeden* interpreter), to update the value of *'x'* from 9 to 12 according to the earlier defined definitions $x = a + b$, where *'a'* is 4 and *'b'* is 5. This is the way in which the dependency maintenance within EM works.

In EM, a script of definitions is used to represent the state of the model. 'State' within this context refers to the set of values of all variables in the model. The only way of leading to a new state is either by redefinition or by adding a new definition. Such redefinition can be done by both human agents and other agents. An example of re-definition by non-human agents is the control of the throttle in the vehicle cruise control model. Here the opening of the throttle is under the automatic action of feedback from the current speed that makes appropriate re-definitions in order to maintain the current speed at the desired value. This feature also allows for an automatic redefinition of the system that can produce an animation of the model. This typically occurs when there are many observables that depend on the system clock.

A simple example of dependency is shown in Figure 2 where `minA` and `maxA` are the angles of the marks on a speedometer dial for the minimum and maximum speed respectively. All the intermediate marks are defined to be equally spaced in between these extremes. So when either, or both, of `minA` and `maxA` are redefined the whole dial is updated to reflect these dependencies. This works in the same way as the automatic update of dependent cell values in a spreadsheet system. This is an essential feature of a definitive script and gives rise to what Sun and Beynon [Sun+99a] describe as its structure of transient values and persistent relations.

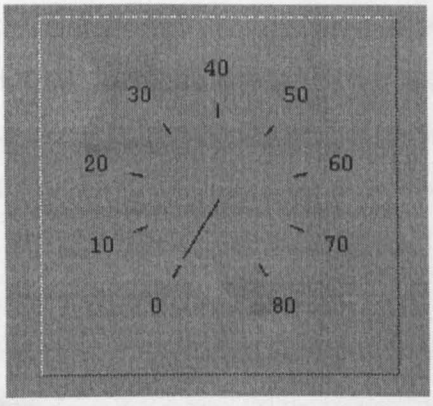
The representation of dependencies by definitions involves operations that include user-defined functions. This property provides both extendable and adaptable system structures to fit well with the users' requirements. So the general form of a definition is

$$x = f(y1,y2,y3,..)$$

where f is a function of arguments *y1, y2, y3,....* Within *tkeden* functions may be written in a way similar to functions in the *C* language. Explicit action or, in short, 'action' is a language feature used in *tkeden* introduced by keyword 'proc' and providing a way of defining a named group of instructions or re-definitions. An example of an explicit action is

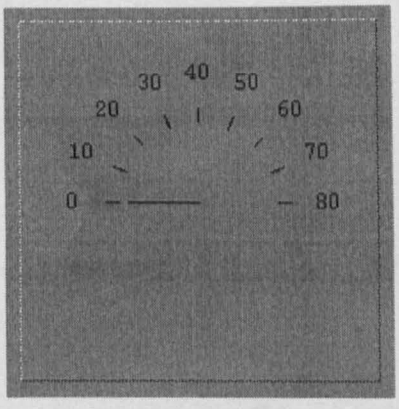
proc display_v: v {display (v);}

The three language features – *actions* (either by an implicit trigger within the computer system or by an explicit trigger from the human agent), *definitions with dependency maintenance* and *user-defined functions* – are provided in a notation Eden and its associated interpreter named *tkeden*. Eden is a general purpose language (a definitive notation) for definitive scripts. DoNaLD is a definitive notation for line drawing (to produce a graphical interface). And Scout is a definitive notation for window management (to provide a visualisation of the model). These three notations can be used freely together for model construction.




```
    :
    real      minA  =  4 * pi  div 3
    real      maxA =  - pi  div 3
    real      topSpeed  =  80.0
    real      A = minA + (maxA - minA) *
                        speedoOnMeter  div  topSpeed
    line      needle  =  [ {0, 0}, {needleLength @ A}]
                  /* the end-point of the needle is defined in
polar co-ordinates */
    :
```

FIGURE 1. A screenshot of a speedometer and its DoNaLD definitions (Extract from the DoNaLD definitions of the speedometer)




```
within speedo {
      minA, maxA = pi, 0.0
                   }
```

FIGURE 2. A screenshot of the above speedometer after a redefinition

In general, the order of definitions does not matter, as they are handled by the *tkeden* interpreter, so that with dependency maintenance, allowing an undefined entity to be included in the script, all the values of observables will always be kept consistent. Table 4 shows two scripts of definitions in different orders which result in identical states. Note that if this same script is written in a conventional procedural programming language, the definitions used in Script no. 1 will not work. This is because procedural programming does not allow undefined variables (shown here by the @ sign), so that the programmer needs to provide the value for each variable first before defining any relation.

TABLE 4. Order of definitions does not matter for state

Script1				Script2			
Definition	a	b	c	Definition	a	b	c
a is 2b + 3c	@	@	@	c=5	@	@	5
b= 4	@	4	@	b=4	@	4	5
c= 5	23	4	5	a is 2b+ 3c	23	4	5
the result is	23	4	5	the result is	23	4	5

However, the order of definitions will usually matter a great deal to the intermediate states during state transition. Table 5 again shows two scripts consisting of the same definition in different orders. While the resulting state is the same in each case the transitions to reach that state are very different.

TABLE 5. Order of definitions does matter for state transition

Case 1				Case 2			
Definition	a	b	c	Definition	a	b	c
a is 2b + 3c	@	@	@	c=5	@	@	5
b= 4	@	4	@	b=4	@	4	5
c= 5	@	4	5	a is 2b+ 3c	23	4	5
b = 7	25	7	5	a is 2b - 3c	-7	4	5
a is 2b - 3c	-1	7	5	b = 7	-1	7	5

The above fact, together with the ease of revision of scripts and interaction with our models allows us to be more relaxed about the planning and development of a

script, compared with the writing of a conventional program. Another consequence of the relatively loose structure of our scripts is that there is a potential for many streams of updating activity to proceed in parallel. [Rasmequan+00a]

The environment for the users of *tkeden* offers some limited assistance for the management and debugging of scripts. However our notations are primarily research tools, sufficient to explore the principles, but far from having the robustness and efficiency required for large scale applications. There is now a distributed version of *tkeden* called *dtkeden* that allows a variety of modes of communication between a server and many client users. In the *broadcast* mode each message from a client goes to the server and is then broadcast to all clients. The *private* mode supports one to one interaction between the server and any particular client. The *interference* mode allows the server to interfere directly in any interaction between clients. In *normal* mode the access privilege of clients reflects management control and social relationships between clients. This offers the potential to support collaborative modelling where many people with different viewpoints and knowledge may interact with a common model and work to achieve consensus on properties or objectives that are initially in conflict. [Rasmequan+00a]

3.3 Theoretical Arguments

3.3.1 The nature of the business environment and the limits of mathematical reasoning

Business studies, generally rooted in the discipline of social science, are human-centred and there are only a few accepted theories that can explain their phenomena compared to those found in pure science such as physics, mathematics and chemistry. On this account, there is a need for a shift in how we try to explain, or speculate about, the phenomena occurring in business environments. Pure mathematics and logic alone cannot cope with such a highly volatile world as that of business. Business environments are considered to be highly volatile because they involve many human factors, and rapidly changing and unpredictable events. For example, a business model for

forecasting a sales volume features *consumer satisfaction* as one of the entities of the model. This kind of entity is rich in meaning and can vary very much depending on by whom and by what means observations are made.

Human factors are not like those factors which are encountered in physical science (e.g. in studying chemical substances in chemistry or bio-molecular structures in biology) which possess fixed patterns of components and behaviours. The pattern of life of an individual is already hard to explain and speculate about. Group behaviour is even more difficult to interpret and predict. This is why it is very rare to find any theory to explain phenomena which are mainly concerned with human behaviour. Evidence for this is shown by the tools used by the economist in studying different influential factors in an economic situation. Those tools are mostly based on mathematics and logic. Such tools are sufficient to prove the mathematical theorems of the economist. But they are insufficient to handle the uncircumscribable nature of real situations. This claim is supported by the fact that, in history, no economic theorem has been able to predict an economic crisis prior to its happening. The economic crisis of Southeast Asia in 1997 is believed to have been caused mainly by the influence of investors (a human factor) in the financial market in Thailand and this led by a domino effect into other countries in the region. This kind of phenomena is difficult initially to give any mathematical theorem to describe it, or its significance, correctly. One reason is that the mathematical theorems of economics are abstract and circumscribed. For such abstraction, the observed phenomena need to be described by fixed patterns that take only limited account of the context in which they are applied. However, when new factors are first observed they are not sufficiently developed to be mathematically or logically expressed and therefore they cannot be included initially. The inability to incorporate the observation in its initial state into the model, means the modeller loses the opportunity to explore further a property that may later become a significant factor of such a model.

The above discussion draws attention to the inherent limitations of mathematical methods for developing models to describe human activities. This may be because human reasoning is too complicated to be able to be fully captured by any mathemati-

cal theory. In reasoning, for example, people involve senses, feelings, emotions and their ego with their objective goals in their judgement. Such things cannot be fully described and included in any theory that tries to predict human reasoning in responding to certain actions or factors. Human reasoning, like many natural phenomena, cannot be fully explained yet by any scientific theory. There is no exact and comprehensive pattern that is fully understood. For example, the precise behaviour of a volcano cannot be predicted yet by any theory.

H. A. Simon has been one of the leading figures in applying computing to management science since the 1960s. He believes that most management problem solving processes can be captured and performed by computer-based systems. This is because – in his view – they can be accounted for by rational human behaviour, whereby humans perform problem solving or making decision based on rational thinking which is most often represented by factual statements or propositional knowledge. This belief contributes to his work in seeking to understand how computers and humans can behave intelligently. In spite of the association of computing with mathematics he made the remark that, *"Many, perhaps most, of the problems that have to be handled at middle and high levels in management have not been made amenable to mathematical treatment, and probably never will ..."* [Simon60, p. 21]. Tactical and strategic problems remain unamenable to mathematical treatment.

The fact that mathematics is insufficient to describe all natural phenomena in general and those that involve human faculties as factors in particular, forms an argument that human-centred activities need a different kind of computer-based support from those in a physical science discipline. Empirical Modelling (EM) is an alternative approach to exploiting the use of computer-based systems to support human activities of this nature, in particular, a business environment. Empirical Modelling as argued by Paul Ness [Ness98], a former Ph.D student of the Empirical Modelling group, puts the emphasis on the 'form' of software over its 'function'. 'Form' refers to the visible part that appears to the modeller. This primarily refers here to the visible structure of scripts. This is because of the higher level of language structure used in an EM environment. With this kind of structure almost all variables have a 'natural' meaning and

much state change (to do with dependency maintenance) is managed automatically. 'Function' refers to the invisible part running within the computer-based system. Ness argued that the emphasis on 'form' over 'function' makes software visible. The significance of the visibility of software is similar from both the users and the developers points of view. That is, for users the 'form' of the interface is how the users have contact with the software systems and this should be something that is visible in the sense of being in a form that is understandable by them. Likewise, for the developers, the 'form' of the interface is how the developers have contact with the hardware systems and this should be something that is visible to them too.

The concern with visibility is relevant to one of the hypotheses made in this thesis that the abstract notions of conventional software development provide less cognitive support compared to those notions of an experiential nature offered by an EM approach. An EM environment provides experience-oriented notions, for example, observables and dependencies. Definitive scripts, in general, are computer-based notions that support the description of phenomena in terms of what modellers experience in their everyday life. By using definitive scripts, an EM environment keeps experience 'alive'. This is the significance of the concept of observables, rather than variables, used by an EM approach that allows the experience of the model, through visualisation and interaction, to be compared with the experience of the referent. In an EM environment, a pictorial-based model is used as a mediation between the computer-based system and the real world referent. Pictorial representation is one of the most effective cognitive aids. In addition, the integrated feature of visualisation, as an inseparable part of the model makes the visualisation of EM models differ from those where visualisation is an add-on feature. Nevertheless, the pictorial representation is neither the only notation that supports human cognition nor the only notation provided by the EM environment.

In a broad sense, conventional ways of computing are based on what Marvin Minsky [Minsky88, p. 329] called 'logical thinking'. According to Minsky 'logical thinking' refers to, *"The popular but unsound theory that much of human reasoning proceeds in accord with clear-cut rules that lead to foolproof conclusions. In my view, we employ logical*

reasoning only in special forms of adult thought, which are used mainly to summarize what has already been discovered. Most of our ordinary mental work – that is, our commonsense reasoning – is based more on "thinking by analogy" – that is, applying to our present circumstances our representations of seemingly similar previous experiences ...". The idea of Minsky about 'thinking by analogy' is among the major concerns in this thesis and being developed as an hypothesis that is given practical, computer-based expression in the approach of EM.

One element of our fundamental philosophical thinking, in the EM approach, is that pure mathematics, which is based on logical thinking, is not the only principle for exploiting computers. As stated by Minsky, logical reasoning is used mainly to summarise what has been discovered. That is, the conventional use of computing that emphasises the application of mathematical and logical statements is suited to traditional data manipulation, or information processing, where situations are well understood and all variables are known, or assumed, and only need computer-based support systems to do calculations, make deductions and make the findings explicit. For situations with a high degree of uncertainty, e.g. the construction of a business strategy plan, a study of social behaviour or a simulation of a new engineering process, we may need a different capability from computer-based support systems. We consider the potential of using computers as scientific instruments in doing computing. In this respect, the consideration of state-as-experienced is the focus and complements the limitation of state-as-abstracted. The following discussion explains these two terms.

3.3.2 Matters of state

In everyday life, we usually exploit both perspectives on state, the state-as-abstracted and state-as-experienced in making sense of our everyday activities. This is where the concern with state-as-experienced becomes prominent in an EM approach to computing. In conventional computing we rely mostly on state-as-abstracted and neglect the significance of state-as-experienced (see Figure 3). The following discussion is concerned with these two perspectives on the concept of 'state'.

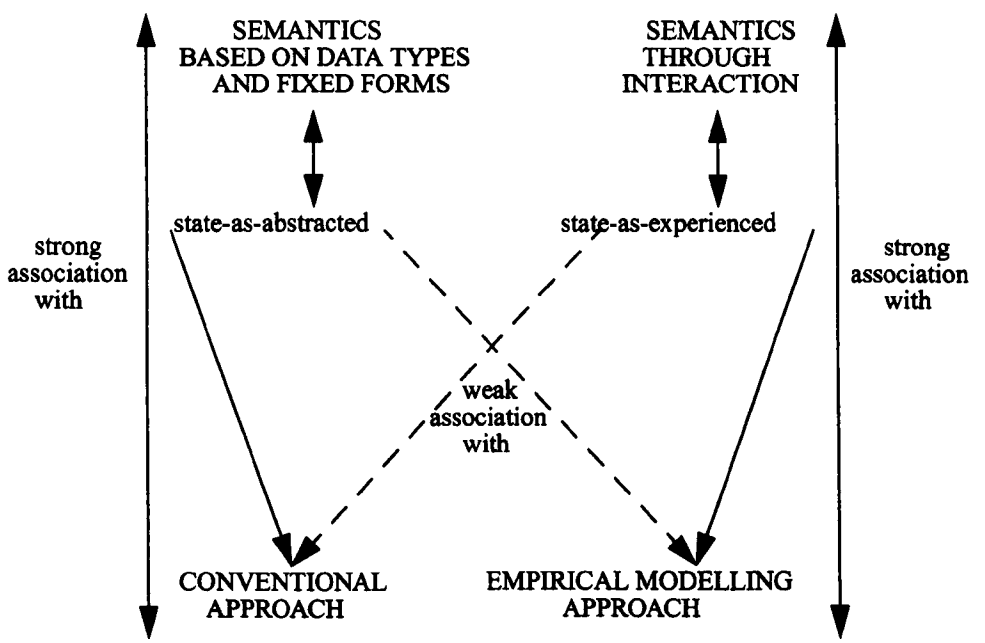


FIGURE 3. State-as-abstracted and state-as-experienced

State-as-abstracted

State, in general, refers to the way things are. Most things are too complex for their state to be fully described. So some form of abstraction is used to simplify the description. State-as-abstracted refers to a particular perspective on state, that is, a representation of the state of things by meanings developed through interaction with them. Such a 'representation by the embodiment of meanings' may not be expressed using a specific and conventional form. Mathematics has a long history of describing state and structure through number systems and algebra. Here the emphasis is on the

description of static systems. For example, data types such as *boolean, real, integer, point,* and *float* are abstract forms that are used to represent real world objects or subjective concepts. Because of their abstract quality, it is easy to perform operations in this kind of state with completely 'unrealistic' results. Such as making the length of a vehicle negative or mis-using the concept of 'average' to say that the number of children in a typical family is 1.25. The change of state is made by changing the values which are limited and depend on the type of the corresponding constants or variables. In science the concern is more with dynamic systems, with behaviour and the prediction of behaviour. This gives rise to another form of selection (or abstraction) where it is the states that are necessary for describing a behaviour that are identified and incorporated in theories and models. Here what is referred to by state-as-abstracted is dominated by what can be measured, e.g. temperature, pressure, wavelength.

Conventions of programming procedures are abstract concepts and are not how things are normally experienced. That is, in thinking or everyday life, people usually neither do events in sequential order nor do them comprehensively. Not all factors can be easily measured. In addition, the practice of prescribing states in terms of computer conventions abstracts away the richness of possible states if they were derived from actual observations. When things are abstract they are usually circumscribed in the sense that the state-as-abstracted is normally circumscribed. It is circumscribed because we need to have a clear picture in mind, or in other words, have pre-determined every possible state. This kind of system will not work when there are any radical new requirements that have not been pre-conceived in a system. Thus it is not suitable for activities where human interpretation is significant and likely to change with people and over time.

A well-defined problem type is one which is sufficiently precisely described to allow an algorithmic solution – i.e. a solution which can easily be represented in terms of computer states. Such a problem type therefore, in a sense, already has alternative answers to select from. That is to say, it is self-solving. Ad-hoc or ill-defined problems can hardly be attempted by this type of programming paradigm. Such problems involve a great deal of human expertise and learning. This is because there may be

few solutions, or even no solution, to such problem offered by conventional systems. Thus it is necessary for human subjects to experiment with computer aids and their own knowledge to explore possible problem modifications and possible solutions. This very process of experimenting enables learning activities which may be a source of new knowledge to be developed. This is because ad-hoc or ill-defined problems are problems for which alternative answers have not been identified yet. There can be no algorithm to derive solutions for unplanned requirements. This suggests the suitability of an open-ended environment to respond to such unplanned requirements.

State-as-experienced in EM

State-as-experienced refers to a different perspective on state, that is, a description of the state of things by meaning developed through interaction with them. Such a 'description by meaning' may not be expressed, or be expressible, in propositional terms. In business, for example, a shop owner may through interaction in a certain way with a shop keeper come to know whether a shop keeper has performed his or her duty efficiently. Such an interaction may be through a verbal enquiry about what is the major enquiry of that day from customers or what is the busiest time of the shop during that day. However, the observation that a shop owner may make is of the body-language of the shop keeper while remembering the events of the day and coming up with an answer. That is to say, the verbal answer is less significant. The feedback from the body-language of the shop keeper is more significant. This kind of observation and interpretation can be made with a certain kind of interaction (being there and observing the physical gestures rather than talking on the phone) and is based on the experience of the observer.

The state-as-experienced literally means the state as experienced by the user or modeller. That is to say, state as observed by the user or modeller during actual interaction with the system. The state-as-experienced refers to the *uncircumscribed, continuous* and *unreliable* nature of the state-of-mind of the modeller or user when viewing the state-of-the-computer and the state-of-the-referent. This kind of state can be represented on a computer by employing an unconventional modelling process together

with the tools called definitive notations. Such a modelling process is based on the construction of cognitive artefacts or, in other words, Interactive Situation Model (ISM)[1], in the form of annotated geometric models which build on the computational models based around relational databases and spreadsheet principles. Beynon at el [Beynon+00a] state, *"An interactive situation model (ISM) is a computer-based artefact that is used to represent state-as-experienced. Its interpretation is not fixed in the mind of the modeller, but is dynamically shaped through interaction with the model, and with direct reference to its associated situation (the referent) ..."*.

Representing the domain with these annotated geometric models, is a way of making 'thinking by analogy' explicit. Such a modelling process also involves consultation with many interested parties and this requires the modelling tool to have a distributed version, such as the *dtkeden* tool already mentioned in § 3.2.2. This way of developing computer-based systems is different from abstract and circumscribed computational models. However, it is accepted that a certain degree of abstraction, albeit empirically established, is also employed in our model constructions.

In applying EM to computing, the significance of state-as-experienced is that, when interacting with computer systems, users feel acquainted with the state of computers as if they were interacting with the real world situation. The EM approach puts a major emphasis on the concept of state. This is because of awareness of the high degree of uncertainty in our knowledge of the world and the necessity of having coherence between the state-of-the-computer and the state-of-the-referent. That is, the flexibility to include new possible states allows a system to cope with uncertainty and achieving coherence between model and referent helps to gain reliability of a system. However, the idea of state in the EM sense differs from the formal concept of computational state. The formal concept of computational state, for example, the state of a Turing machine or a state-chart, has a *pre-defined, discrete* and *precise* nature. These properties of state are typical of state-as-abstracted. This formal concept of computational state pays attention only to the state-of-the-computer. The EM concept of state

1. A term introduced by P. Sun, a former PhD student in the Empirical Modelling group [Sun99, Sun+99a]

is broader. That is, the state under consideration includes the state-of-the-computer and the state-of-the-referent in coherent relation. The quality of coherent relation is judged by the state of mind of the user, as they experience it. This is the origin of the term state-as-experienced in EM as opposed to state-as-abstracted by the formal concept.

For example, an airline booking system will implicitly force the user to interact with the system in a pre-defined pattern. That is, the user has to supply at least, either the exact or estimated date of arrival and departure and, of course, a destination, to be able to get a proper response from the system. In addition, the data supplied to the system needs to be of an exact type and structure. It may find that such specific data type and structure requirements are not well matched with their real world experience. For instance, most of the on-line application forms of UK origin will require the user to fill in a field called 'county' and those of USA origin will require a response to a field called 'state' in the applicant's address section. This type of request makes almost no sense to most of the Thai applicants where neither 'county' nor 'state' applies. Such a mismatch can cause minor confusion and annoyance to the user. A more serious problem is the mismatch is the case of the mismatch between the conventional system structure requirements and what the modellers know at that particular time of system development.

In a formal method of computing, all programming procedures, regardless of their levels of complexity, can be reduced to a combination of control sequences and selections or repetitions of basic operations. These types of system structures need complete knowledge of the systems which in most cases is difficult to acquire, and, even when it is acquired, changes in the situation may make the knowledge invalid. This situation leads to a number of failures in computer-based system development. For example, the developed systems do not serve user needs well, or the systems are completed far too late and are therefore useless. Such difficulty leads to the area of interest called 'requirements engineering' which deals with how to capture the user requirements to successfully develop systems in response to such needs. However, the

'requirements engineering' approach to capture user requirements is still applicable successfully only to those systems with well-understood needs.

State changing in a business context

Business activities are context sensitive and situation based. This is because they involve a variety of complex human factors. For example, consumer demand may depend on consumer behaviour which consists of, for instance, an attitude toward a particular advertisement campaign, peer group affection and their personal outlook or their financial status. These kinds of factor vary from person to person, from community to community and there is no theory to explain the variation. In addition, there are, for example, major business problems:

i) the difficulty of maintaining continuity and stability; e.g. nobody can guarantee that the price level of any particular stock in a stock market will continue to move in the same direction as before;

ii) new technologies and working practices are common; e.g. e-commerce, computerised stock control, 24 hour service regimes;

iii) unexpected and irrevocable events are commonplace; e.g. loss of key staff, mergers, a financial crisis, and

iv) patterns of activity that are stable can be influenced by factors that appear unrelated; e.g. a sales routine that has been highly successful may fail as fashions change.

These kinds of problem are due to the inherent character of business activities: lack of theory, unpredictability, unexpected changes and unexpected influences. This discussion shows that the conventional ways of totally excluding individual experiences of the referent lead to a loss of the significant properties of the nature of business activities. Such properties are, as mentioned earlier, context sensitive and situation based.

The process of observing, taking an action in response to such observation and appraisal of the outcome of such action to determine another action, in a particular

context and situation, is essential to bring about the reliability of such a business activity. This process needs a special environment that enables the dynamic change of both data and structure in accordance with such observation. Therefore the abstract pattern of business processes used by most current computational practice is inappropriate either to prescribe or to describe such events. Neither the use of a formal state-transitional model (e.g. statechart or Petri Net) from a traditional computing perspective, nor the use of a collection of rules to define the roles of human participants from a management perspective, can engage with the actual experiences involved in carrying out the process. This is because both abstract computational states, and business process rules, refer to interactions and situations that are presumed to be so well-understood that the observation and actions to which they correspond can be specified without considering their particular context. Such presumptions are clearly not applicable to the nature of the business environment which is affected by a particular context, and each context gives a different outcome [Beynon+00a].

In preparing a sales forecast, for example, the sales volume depends not only on the sale price, but also on the demands of the market, consumer behaviour and the economic situation of that particular market. So marketing planning cannot rely on sales forecasts made only on the basis of the cost of production. The factors mentioned earlier need to be incorporated into the models as different contexts will give different results. The issues discussed here argue for the need for the open-ended environment of EM for business activities. The use of the EM approach, where the understanding of state is broad enough to include state-as-experienced to complement state-as-abstracted, allows the inclusion of information derived from observation on the scene as the activity proceeds. The flexibility to change either the structure, or the data, of business models while observing the business in operation is significant. This is because the greater the flexibility of a firm to change the operation of their business models to correspond with the actual situations, the greater the chance that a firm can come up with an up-to-date support system that represents accurate information or knowledge.

The situational and dynamic nature of human activities, cannot be described by explicit and abstraction-based theoretical scientific views of the state of physical systems. For example, the factors that will shape a decision by a manager to launch a new product or a customer to buy a new product are varied and depend on the individual's experience and knowledge of the product and the situation and context in which such a decision is made. In addition, these variations are definitely different from one person to another. Thus a model that is used by a manager of one firm to study consumer behaviour may not work with another manager of a different company or even within the same company. Open-ended interaction is used along with knowledge of the referent so that the interpretation of the models as represented in a script is associated with the modeller's personal knowledge of the referent. This is one of the essential reasons for the claim that EM computer-based models are a form of knowledge representation (see Chapter4). Open-ended interaction enables observation of a dynamic and situational nature to be incorporated in the models.

3.3.3 The use of cognitive artefacts

Norman's introduction of the term

According to Norman [Norman91], who first introduced the term, 'cognitive artefacts', refers to man-made devices that *"maintain, display, or operate upon information in order to serve a representational function and that affect human cognitive performance ..."*. Norman examines cognitive artefacts according to:

i) the system view and the personal view of artefacts (their role in enhancing cognition);

ii) their levels of directness and engagement;

iii) their representational properties.

His work is based on his interest in the power and importance of culture and artefacts to enhance human abilities, and it is focused on the information-processing role played by physical artefacts upon the cognition of the individual. His work in this

area follows the tradition created by people in the field of human-computer interaction: that is the formal study of the cognitive relationship between a person's activities, the artefact of the computer, and the task. However, Norman's work is focused particularly on the details of the 'interface' between the person and the computer. He aims at deriving a scientific understanding of the role of artefacts, through the study of the properties of the artefacts and how their design affects the person and the task. Norman argues that artefacts do not change the capabilities of a person, they only change the nature of the task. He states that once the informational and processing structure of the artefact are combined with the task and the informational and processing structure of the human, the cognitive capabilities of the total system of human, task and artefact are expanded and enhanced.

There are two views of the cognitive artefact: the *system view* and the *personal view.* According to the system view, the artefact acts as an interface between a person and a task that enhances cognition, and expands the functional capacity of the task performer to enable a task to be accomplished. From a personal view, the artefact is an interface that will change the nature of the task from its original to a different one. However, Norman points out that from all points of view, the artefact changes the way a task gets done by: distributing actions across time (pre-computation, e.g the planning of what should be included in a pilot pre-flight check-list); distributing actions across people (distributed cognition, e.g. the distribution of knowledge from one generation to another) and changing the actions required of the individuals doing the activity.

The second aspect of cognitive artefacts concerns the levels of directness and engagement in the use of an artefact. When we use an artefact to do a task, it is necessary to make use of a representation. According to Norman, artefacts act as mediators between persons and the world, in both execution and perception. The execution is between the actions and the resulting changes to the world state. The perception is between changes in the world and our detection and interpretation of the state of the world. Norman discusses varying degrees of interaction, from a very direct engagement to a remote form of interaction. He points out that some interactions are so indi-

rect and remote that feedback and information about the world state are difficult to obtain, possibly delayed in time, and incomplete or of unknown accuracy. These varying degrees of interaction and their consequences can affect the task performance and they are controlled by the design of the task and the artefact.

Norman further points out by referring to the work of Bødker that artefacts can be used to mediate directly between the person and the object and between the person and the virtual object or world. In the second case, they involve different layers of representation which have different formats. Norman argues that, *"The choice of representation and interactions permitted by the artefact affects the interaction of the person with the object, whether real or virtual... Different forms of artefacts have different representational implications, which in turn dramatically affect the interactions ..."* [Norman91].

Norman states that artefacts that are used to support actions must support both the execution and evaluation phases of the action cycle, which usually have different representations. He points out the gaps between the world and the goals occurring during both execution and evaluation phases. He claims that these gaps exist because of the mismatch between our internal goals and expectations and the availability and representation of information about the state of the world and how it might be changed. Norman suggests ways to handle these gaps. Firstly, by having a proper design of the artefact and secondly, through mental effort and training.

The third aspect of cognitive artefacts discussed by Norman is their representational properties. Norman points out that, *"...the power of a cognitive artefact comes from its function as a representational device ..."*. With reference to some of his other work and the work of others, he suggests that a representational system has three essential ingredients:

1. the represented world, that which is to be represented;
2. the representing world, that is a set of symbols and
3. an interpreter, which includes procedures for operating upon the representation.

Norman classifies the level of representation of artefacts into two classes: *the surface representations* and the *internal representations*. Those artefacts that fall into the surface representation category are devices that are primarily systems for making possible the display and maintenance of symbols. Artefacts falling into this category are the physical part of the physical symbol system. The symbols are maintained at the visible surface of the device and serve as the interface. Artefacts that possess internal representations, in which the symbols are maintained internally within the devices, need interfaces that transform the internal representation into a surface representation that can be interpreted and used by the operator. This is because the internal representation is inaccessible to the user.

When artefacts have only surface representations in which the representation itself serves as the interface, the style and format of the interface determine the usability of the device. Norman points out that for an artefact to be usable, the surface representations have to correspond to something that is interpretable by the user, and, the operations required to modify the information within the artefact have to be able to be performed by the user. He further points out that the interface serves to transform the properties of the representational system of the artefact to those that match the properties of the person or user. In Norman's view, for the user of an artefact, the representing world is the surface of the artefact (the information structures accessible to the person employing the artefact). He argues that one of the basic issues in developing an artefact is the choice of mapping between the representing world and the represented world (or between the surface representation and the task domain being supported by the artefact). The choice of representation determines how faithfully the match is made between the represented world and the representing world.

Commentary on Norman's view of cognitive artefacts

The investigation of cognitive artefacts by Norman concludes that their 'critical components' are: their role in enhancing cognition, the degrees of engagement that can be experienced and the role of their representational format. Such components are highly desirable features in computer artefacts but they have not yet been achieved in significant measure in conventional systems. An EM way of using computers where the emphasis is on their ability to enhance human cognition and engaging activity between computers and human subjects, supports regarding computer artefacts in an EM environment as cognitive artefacts in line with Norman's analysis. However, throughout this thesis we shall use the term 'cognitive artefacts' in the very general sense of something man-made that serves some knowledge-related function.

In EM, models can be regarded as cognitive artefacts because of the quality of their state representation. Such a quality supports human cognition. Norman refers to cognitive artefacts as artificial devices that enhance human cognitive capabilities. However, Norman emphasises that the artefacts do not actually change an individual's capabilities, but they do rather change the nature of the task performed by the person. However, while sharing his view on the point that cognitive artefacts could enhance human cognitive capabilities, there is a sense in which this can also mean enhancing individual capabilities. A variation of our view from Norman's is that we believe that cognitive artefacts developed in EM allow us to reinforce our experience and support the learning process. In the modelling process, one can have pre-experience or gain a kind of 'compressed' experience. Once we can reinforce our experience, we often develop a better understanding of things or situations we are dealing with. This is the sense in which we can regard cognitive artefacts in EM as a means of enhancing one's capabilities. EM considers that, with the capability to improve and expand human cognitive abilities, cognitive artefacts support and enhance an individual's capabilities to perform a task better. In contrast, Norman believes that cognitive artefacts just change the nature of the task.

The following quotation by Norman demonstrates a very similar view to that of the EM approach, with the emphasis on cognitive artefacts, *"When the informational*

and processing structure of the artifact is combined with the task and the informational and processing structure of the human, the result is to expand and enhance cognitive capabilities of the total system of human, task, and artifact ..." [Norman91]. This point of consideration is established in applying the EM approach based particularly on the way of constructing computer-based support systems which is parallel to the way that engineers design devices to support human activities in accomplishing particular tasks. Particularly attention is paid to making the systems visible to their users, similar to the way that engineers view their products as they are being designed. With this thought in mind, we believe that the visibility of the system being developed would help promote the cognitive capabilities of humans, as paper and pen has done for most of people since they were first introduced. By and large, the end result of development of this nature is a more understandable. This is because the system is constructed using a personal understanding of the computer-based system closely correlated with real world referents experienced by the users themselves.

The EM modelling process enables the exploration of possible alternative features and functions of the system. This is just like the way engineers try out different features with their products, or managers try out different *what-if* analyses of their financial planning using spreadsheet modelling tools that help them visualise aspects (e.g. the changing of the benefit in response to different cost allocations) of their abstract ideas about financial figures. There is no pre-defined procedure to regulate any possible choice of interaction. The open-ended interaction environment is often where learning and knowledge grow. Unlike modelling, programming principles that encompass the current practices of system development impose a strict route to be followed, and do not allow much exploration.

3.3.4 The modelling process as an alternative means of constructing business support systems

Modelling is an activity where different alternatives for problem construction and supplying of solutions are being observed and analysed. Because of this nature, the personal element in modelling is vital. For instance, modelling may be guided by assumptions made by an individual which are rather personal and depend on individual experience of the domain. In addition, since business activities have more human and social factors than many other activities being modelled, the development of business models needs to consider the incorporation of human and social aspects into the models to a greater degree. Rivett [Rivett94] points out that model building is unlike constructing a mathematical solution in the sense that there is no single correct answer.

Business support systems, for example DSS, are a means whereby a manager can interact with a computer-based system to get supportive data and information. Business DSS is a particular system that incorporates data and models to provide supportive feedback for business decision makers in a specific problem solving situation. However, the success or failure of a decision does not rely solely on the procedure by which the decision was made, but also on all the assumptions made during the modelling process. For example, a production manager has to make a decision about whether to expand his or her plant production capacity to serve increasing demand. For this purpose, the manager needs to make the right assumptions regarding the real situation of the production capacity and market demand. For instance, it might be the case that there is a temporary lag in production, rather than insufficient production capacity. Thus the modeller has to be cautious in deciding proper assumptions when forming a decision model.

The issue of 'hidden' assumptions in a model is a major problem. In a specialised design support system, all the assumptions that a model is based on are largely out of sight from the users. This makes it both difficult to alter if there is any sudden and radical change in the model or to understand properly on what grounds such a model

has been derived. However, spreadsheet systems seem to alleviate these problems quite well. Users can always check back to what sort of assumptions have been made in a particular model by checking the formula and an instant change can be attempted. Spreadsheet systems introduce a new direction of including human ability into the use and the development of computer-based support systems. Beynon et al state that, *"The application which has perhaps been the greatest single influence for enhancing the quality of human-computer interaction is the spreadsheet. The interaction the spreadsheet allows, and the cognitive engagement with the user that it enables, has no conceivable counter-part within the batch mode of computation of the 1960s ..."* [Beynon+02, p. 9].

Decision models needing proper assumptions, and the progress of a development environment for DSS, draw attention to the fact that neither a generic system nor a specialised system fulfils the real needs of the business environment. The generic package treats every firm's activity of the same kind as being the same which is most often not the case. Different firms have their own individual practices for the conduct of their business. The generic system is 'too-broad' to fit well with what the user wants. On the other hand, a system that was specially designed by a system expert for a particular firm is 'too-specialised' which is sometimes found too difficult and complicated for the users. Users expect something that is just 'about-right'. That is, they aspire to 'end-user computing'.

Business decision modelling within this context refers to the building of a model that represents a particular real world circumstance which is confronting a manager. Then a particular issue or issues through the modelling process will arise for decision taking. Via the interaction with the models, managers get support to figure out the anticipated results if particular factors are altered in different ways.

Modelling process for supporting unstructured problems

Solving unstructured problems, such as the work of strategic management, is one of the factors critical to the success of business operation. In an unstructured problem there is always a number of things left for speculation, interpretation and debate. Strategic management here refers to *"... the reading of signs and portents of the future and*

interpreting them in order to choose an appropriate direction for the future development of the organization ..." [Robson97]. This statement indicates that business activity of strategic management is unstructured by nature. This is because this activity mainly deals with things in the future which cannot be perfectly prescribed or predicted. It deals with speculation and interpretation, where each individual has his or her own speculation about, and interpretation of, a particular thing in a particular circumstance. To be able to speculate properly about the future, a good understanding of the current situation is crucial. The modelling process gives a good support to understand things or situations better.

Dealing with things in the future and being based on individual interpretation would already be enough to make such problems difficult to identify or solve. In practice, the development of a decision model for this type of activity is mainly done by a business executive's thinking with the typical cognition aids, paper and pen. This type of decision model is developed within the mind and is based on previous experience, together with an external aid in the forms of data or information retrieval and manipulation e.g. a calculator or a computer. That is, in spite of the intelligence with which a particular human brain can work, the use of a computer-based support can still assist the process. A computer-based system that supports the construction of models of this kind of problem could help managers in explicitly identifying the concepts and constructs that they use when thinking about their business. This would lead to a more solid structure of ideas which would enable the manipulation of alternatives to be more efficient.

A computer-based system that enables the building of models which highlight the influential factors of a business, would be an enormous support for any firm. The models produced would exhibit the knowledge of those executives about particular areas of their expertise. This is good enough to use as a supportive aid in making important judgements and decisions rather than leaving intuition alone as a dominant decision-making component. Those computer-based models which are considered to represent particular business knowledge can be used as a medium to transfer that specific knowledge from generation to generation. However, no models, whether built

solely by the human brain or by a combination of computer-based support system and the human brain, can guarantee a right decision.

Strategic management activities, for example, strategic analysis (i.e. expectations, objectives) and strategic choice (i.e. generation of options, evaluation of options) are in need of a powerful tool to construct a decision model which combines quantitative and qualitative elements. The quantitative elements are those commonly found in a well-structured model while qualitative elements are typically based on the experience or knowledge of an individual. For example, in the case of the marketing department mentioned above, in addition to the numerical component (quantitative issues e.g. a monthly financial analysis or a weekly machine utilization report), the addition of a non-numerical component (qualitative issues e.g. level of staff satisfaction, the potential of a new market, the extent of management's fear of his or her competitors) to the decision model could be a supportive tool for management in strategic decision making.

The quantitative element, as the name implies, can be measured in terms of some unit which can be used directly in the modelling process. Pure science or 'hard science' like physics, chemistry and engineering has enjoyed the benefit of being able to construct and manipulate models with such elements for centuries. On the other hand, the qualitative element which is quite vague and personal but often crucial to managers in decision making, has long been a pitfall for social science or 'soft science'. However, this does not mean that qualitative measurements have not been attempted. Most common qualitative measurements are by ways of quantitative comparisons. Scholars in various disciplines have developed scales for transforming a qualitative element into a quantitative element for further processes. For example, in a questionnaire to survey the level of satisfaction of consumer on a product, a number from 1 to 5 might be used to represent the level of satisfaction. So a company can further process those data, to find out whether the level of satisfaction will effect the purchase of such a product or not. Such transformation of qualitative elements into a form of quantitative measurement may have to continue unless there is a breakthrough in measuring qualitative properties.

Within the domain of unstructured problem situations, such as those involved in strategic decision making, the construction of decision support systems requires a special kind of modelling process. Such a kind of modelling process needs to support the process of knowledge acquisition or the development of insight. This is because the problem is unstructured in nature. There is a great uncertainty about what structure a business 'master plan' should have. This is due to the tremendous number of unknown factors. For example, the decision makers will never know for certain what the future situation will be like or what are the actual factors that will shape such situation. They have to make a number of alternative assumptions based on their best knowledge of the field which is the result of their education and working experience. After arriving at a number of assumptions, the modelling process becomes a significant instrument in exploring both quantitative and qualitative data and the potential consequence of each strategy.

3.4 A Proposal for a Paradigm Shift

3.4.1 The need for a shift in paradigm

From the theoretical arguments made above, attention has been drawn to the need for a shift in the paradigm for the use and development of computer-based systems to support human activities. The term 'paradigm shift' was introduced by Kuhn in 1962 [Kuhn70]. Kuhn suggests that any scientific discovery is the result of the application of unconventional standards of practice. He emphasizes that, without different outlooks, new significant features are hardly found. Our use of the term 'paradigm shift' here is in line with that suggested by Kuhn.

Conventional ways of using computer-based support systems impose constraints because of the need to circumscribe the system comprehensively and to restrict the interaction between the human agent and the computer-based system in pre-defined ways. Such interaction resembles playing a typical conventional racing-car game on the computer. That is, there are certain sets of events to act upon and there are solutions embedded. Within this kind of environment, the possible states of the computer

are circumscribed as a result of employing programming conventions. There is little chance of exercising new interactions which arise from a new discovery or idea. This could happen with an environment that allows arbitrarily detailed modelling which would enable the user to experiment with a new discovery or idea. Through arbitrary interaction an unusual result may be observed which may lead to new interpretations of observations or understanding of the domain. An environment for doing open-ended interaction is not really possible within current computing paradigms. This is because such an environment needs to be able to entertain all possible states derived from observation of both computer systems and referent systems, like a live debate where a participant waits for an unpredictable response from another party to be able to respond properly. Such an environment is uncommon because there are too many 'possible states'. It requires a lot of involvement from human subjects to operate this kind of system which goes against the current tendency to maximise the degree of automation.

Consideration of possible new directions for computer use and development invites a shift in computing paradigm. An alternative framework is suggested here which exploits the use of computer-based support systems for business. Conventional approaches are usually conceived within the method-tool-user framework [Beynon+00a] shown in Figure 4. Figure 4 shows the recipes for action that are associated with the tool, and a family of interactions with the external world that is reliable and repeatable. Both set of activities represent the involvement of human agency. Carrying out the procedures on the left-hand side demands skill and knowledge in the use of the tool. Usually the users are occupied in corresponding activities in the external world. There are well-established protocols that link interactions with the tool and interaction with the world. This practice has probably been influenced by dominant concepts of automatic devices adopted before electronic computers and during the early period of computer-based system development. In such a framework, human-computer interaction is framed in terms of three interdependent concepts: methods, tools and users.

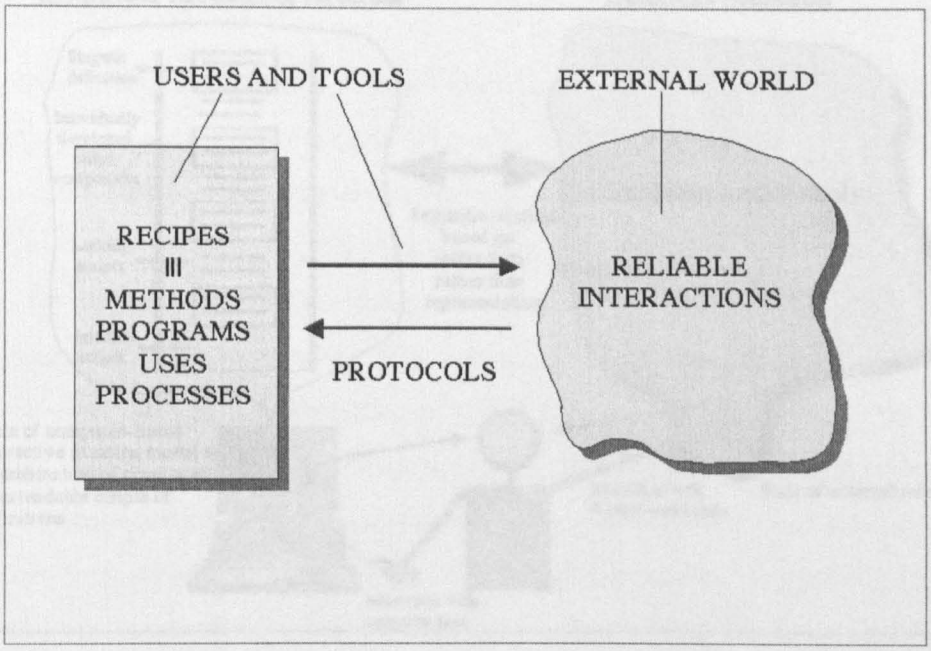


FIGURE 4. The method-tool-user framework

The computer-as-agent framework, proposed by EM and shown in Figure 5, offers an environment for open-ended modelling with scripts. Within this framework, the idea of fully integrating human and automatic activities is promoted. This framework is suggested in response to the theoretical arguments made earlier:

i) the prevalence of state-as-abstracted in conventional programming (due to the influence of the success story of mathematical and logical statements in describing physical phenomena) is unsuited to describing business phenomena where the significance of state-as-experienced is crucial;

ii) the need to use computer-based systems as cognitive artefacts as instruments to explore a new territory of interest or problems as opposed to the use as an automatic information generator that would largely summarise findings or generate guided discovery;

iii) the need to use simulation modelling as a method for developing systems to serve specific purposes.

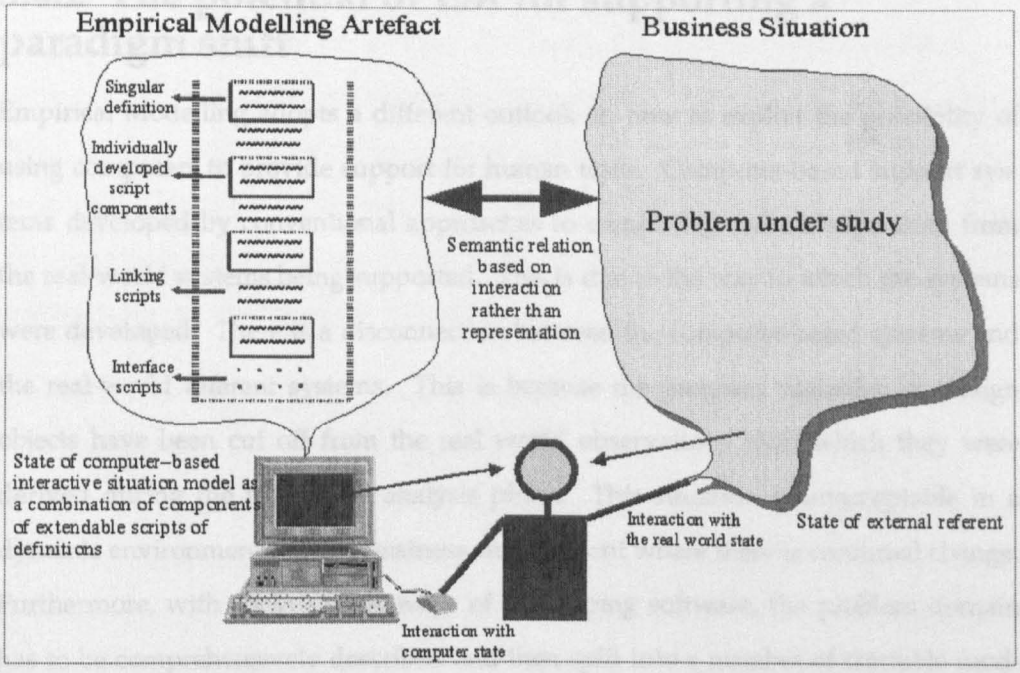


FIGURE 5. The computer-as-agent framework for open-ended modelling with scripts

Figure 5 depicts the semantic relation as an association between the state of a computer-based artefact on the left hand side of the figure, with the state of the referent or the business situation on the right hand side. The open-ended script reflects the possibility for unbounded extension of the family of observables involved in establishing such an association. The framework of computer-as-agent shifts the focus of computing from behaviour that can be dictated by global rules, to interaction that emerges from local stimulus-response patterns. Along with this shift in focus is a corresponding change in the pattern of use of the computer-based system from a tool for calculation and storage to an instrument for processing and communicating knowledge.

3.4.2 The potential of EM for supporting a paradigm shift

Empirical Modelling adopts a different outlook on how to exploit the possibility of using computers to provide support for human tasks. Computer-based support systems developed by conventional approaches to computing cause a separation from the real-world systems being supported. This is due to the way in which the systems were developed. There is a disconnection between the computer-based systems and the real-world referent systems. This is because the program variables, or design objects have been cut off from the real world observations from which they were derived during the traditional analysis phase. This situation is unacceptable in a dynamic environment like the business environment where there is continual change. Furthermore, with conventional ways of developing software, the problem domain has to be comprehensively described and then split into a number of tractable modules to be repeatedly coded and tested against their required properties. After that, these individually developed units will be combined and made ready for implementation. Such a practice encounters a number of difficulties.

Firstly, even if possible, it is a difficult task to describe comprehensively any problem that is based in the world of everyday affairs. This is because no-one is capable of knowing the physical world completely. We gradually learn from experience and understand things better through our interaction with things. From time to time, we may need different kinds of support, as we develop more skill in our work assignments. Secondly, it is always the case that due to the constraints imposed by conventional programming, each component of the system has to be built according to the limitations of the language used. Sometimes such a practice can cause the loss of significant features. This arises because of the improper substitution of one feature for another imposed by such language limitations. For example, an interior design piece of work which is using a built-in function for drawing a circle to draw an oval shape of a particular piece of furniture, may (if the designer does not make it clear enough that the shape is really an oval) mislead the customer to presume that such a piece of furniture will be circular. Thirdly, the separation of an identified problem into indi-

vidual parts and the reassembling of these after coding can cause a lack of coherence in the overall structure of the system. Finally, the developed system has been transformed so much from the original by the domination of mathematical or logical rules that it cannot faithfully reflect either the real-world situation or the real need of the users. EM offers a distinctive way of constructing a computer-based support system. The main consideration is to develop a support system that better reflects what the users actually experience in the real world.

With the use of the EM approach, the causes of those typical difficulties discussed earlier are avoided. We introduce systems built in an incremental fashion to avoid the difficulty in comprehensively describing the problem domain. It is not only this feature of incremental development that enables us to avoid such a difficulty, but also the feature of redefinition that enables us to add in, or alter after development, any system feature that we consider better suits the system, with little extra cost. We offer an alternative way of describing state called 'definitive scripts' (see § 3.2.2) to avoid the limitations of programming languages. Then the way the system is built using an EM approach combines traditional phases of system analysis, design and implementation so that it helps to avoid the problem of lack of coherence (see § 4.4.3).

Using the EM properties discussed earlier, computer-based support systems can be built which incorporate the 'cognitive artefacts' discussed by Norman (see § 3.3.3). These cognitive artefacts, together with their distinctive quality of human-computer interaction, make such EM type computer- based systems into a rich kind of knowledge representation system (see § 4.3). However, knowledge representation, in our sense, is different from context-free symbolic representation in a conventional sense. For example, in developing an entity-relationship model, the use of text and diagrams to represent the software system and its behaviour, is a context-independent type of knowledge representation. EM offers a computer-based knowledge representation system in the sense that a system is built through an understanding of the problem domain by the modeller's interpretation which is a context-dependent activity. This context-sensitive activity also makes EM computer-based systems different from many of those in the current practice of Artificial Intelligence. Winograd and Flores

point out that the design of an intelligent system has restricted knowledge representation by the acquisition and manipulation of the adopted facts [Winograd+86]. The same authors' argument that knowledge is *"... always the result of interpretation, which depends on the entire previous experience of the interpreter and on situatedness in a tradition ..."* supports very well EM's emphasis on experience and interpretation through interaction. This gives knowledge a contextual nature and explains how EM computer-based systems can possess the capacity for a particular kind of knowledge representation.

Within the EM approach, computer-based support systems include the computer as an instrument rather than an invisible machine [Norman98], and humans as significant components of the systems (including their experience and interpretation ability), together with the context (both internal environment – computer-based system and external environment – a real-life situation) within which the operation takes place.

EM's cognitive artefacts and their roles in supporting human activities

The use of cognitive artefacts. Payne [Payne99] expresses a similar view to the one introduced in this thesis that a computational system can be a kind of cognitive artefact, or tool of thought, to support everyday human problem solving. Common kinds of cognitive artefacts, according to Payne, are pencil-and-paper, diagrams and graphs. Payne's idea of computational systems as cognitive artefacts that support human everyday problem solving, is in line with that of Norman [Norman90]. The EM approach regards a computer-based artefact as a representation of state. State, in the EM sense, refers to the state of a single model that represents a common feature of the state of a real world referent, the state of the computer system, and the state of mind of the modeller. In this sense, representations of state are based mainly on the modellers' personal knowledge of the domain. Models of the same domain problem may therefore vary because different modellers will have had a different experience of the world and will have obtained different knowledge about it. This is one of the fea-

tures of an EM model as a form of knowledge representation. However, since a computer-based support system is generally aimed at supporting a group of people working together, the representation of the system needs to achieve public consensus before it can be applied.

In this usage of computer-based support systems, state is the main focus initially because state shapes the consideration and interpretation of a particular computer-based support system at a particular moment in time and one context will differ from another. A particular state of a computer-based support system is a representation of specific phenomena at a specific point in time in a specific context. Because of its use of definitive notations, the representation of a system by a notation in EM is normally expressed by observables and dependencies which are in terms of the common understanding of people in the same domain. By this kind of representation, the current domain knowledge of the field may be transferred from generation to generation. This is another attribute of the EM model that enables computer-based support systems to have the property of knowledge representation.

An EM model is considered to work for a system developer in a similar way, for example, as a mechanical model works for a car design engineer in prototyping a new car model, or as a molecular model works for a chemist in analysing components of a certain substance, or as a construction model for an architect in designing an exhibition centre. These types of models have a significant common feature, the quality of being guided by visualised interaction. These models act as media for developers to represent physical characteristics explicitly either by material objects or by pictorial images of the domain under consideration. The developer's interactions with the model via open-ended interactions can be considered as a representation of the modeller's subjective construal of the domain. These two kinds of representation – of the domain and of the modeller's view of the domain – enable experience.

The combination of the physical appearance (i.e. a concrete object representation like a mock-up of a car model, a subjective concept representation like a textual or pictorial visualisation or a computer model of a market analysis model), and the dynamic

interaction with such a concrete object, or subjective concept, are well suited to the cognitive processes of humans when they are learning about a new domain and need to pay attention to them (in order to get to know them). For example, in explaining to a student how to solve an equation, a teacher has to write it down on the board (giving the physical appearance by means of visualisation of the subjective concept) and then verbally explain and discuss it with her students (having a dynamic interaction between teacher and student). Another example is when we first learn how to operate a washing machine from a manual, which in most cases will contain a pictorial image of the machine and indication of major components that a user needs to know. Although a verbal explanation may give some indication of how to operate the machine, actual physical contact or interaction with the machine is necessary in order to relate this knowledge to the machine and operate it.

The direct physical contact with the car (concrete object), for example, or an indirect physical contact via the mouse click and keyboard with the EM model, enables a strong interaction between the human agent and the model she is working with. It is evident that whenever there is direct physical contact with something of interest, it is likely that the involvement will continue for a longer period than when this is lacking. In the case of indirect physical contact this is less clear. Some might argue that conventional modelling, using mathematical and logical arguments, can retain the modeller's engagement. This may be true, however with the EM model there is a different level of engagement between the modeller and the model because of the unusual nature of EM models. The structure of the EM model is based on definitive scripts that can embody the common understanding of people working in that area. In addition, the visualisation of the model represented by pictorial or metaphorical icons supports a feeling of having physical contact with the model, which a purely mathematical or logical model alone could not offer. When interacting with the EM model, system users who are not experts in mathematical or logical programming may find themselves having a familiar interaction with the computer models as if they were interacting with real world referents more than they would do with conventional types of model. In addition, the way the phenomena are described by definitive

scripts which define the relationships of the observables, makes the EM model a sense-making one (see § 4.4.4).

The open-ended interaction of the EM approach overcomes the limitation of the preconceived environment of conventional models which only reflects viewpoints or scenarios pre-regulated to govern the way things may work within that particular system. In such a system, the interaction is implicitly restricted. For example, a stochastic simulation model for a business analysis has already embedded in it a certain number of preconceived types of interaction. These need a set of inputs of given types that must be acquired either as a real data set, or an estimated data set, that has to be entered into the model to be able to run the simulation. Using EM, a similar model may run without the complete set of data and this needs a different way of manipulating the available data. This situation, where data or information is incomplete and requires human interpretation, reflects very well the real nature of the business world and our daily life activities.

Roles of cognitive artefacts to support human activities. Artefacts are built to facilitate and enable the enhancement of human capabilities. For example, vehicles are created and improved to enhance and extend human capabilities for travelling, e.g. to travel faster or to travel a longer distance, that is, from carts, trains, cars, aeroplanes to, a recent means of travelling; spacecraft. These artefacts have been developed to cope with people's desire for vehicles to match perceived need.

Computer systems are one of the artefacts that can be used to enhance human capabilities. The memory storage of a computer is greater than that of an ordinary person's brain. For example, in his well known work [Miller56], Miller states that seven, plus or minus two, is the maximum number of items a person's brain can handle at one time. This limit makes the assistance of computer memory necessary , for example, in dealing with situations that involve more than seven factors at the same time. This ability to handle items of factual data must not be confused with the human capability of retaining a vast amount of varied common sense knowledge and the ability to make a reasonable judgement based on such knowledge. The calculation

power of a computer is much greater than that of human beings but this is not true of other computational abilities. For example, the human power to recognise images or visual representation or to learn language still far exceeds that of computers.

One of the motives for the EM approach is to introduce computer-based systems to support human everyday problem solving. With this focus in mind, a real world situation is chosen for study and a computer-based system is constructed to operate it or to solve its problems. To do this the EM approach emphasises the use of a rich representational system: a system of cognitive artefacts as opposed to a system of mathematical abstractions which would be created by conventional approaches. The EM representations have some similarity with what Johnson-Laird called 'rich symbolic systems' which have an infinite number of possible symbols, they are, for example, architect like drawings, mathematical notation, alphabetic writing [Johnson-Laird93, p. 29]. They also have *physical contact enabling interactions*, i.e. the monitoring and controlling of the systems through the mouse click interfaces. This combination leads to a sense of real-world interactions, instead of the purely abstract interaction of the conventional approach in which only interactions between abstract command interfaces are allowed. For example, in the restaurant management model (see § 5.4.1), an EM environment provides a stronger form of cognitive aid than the normal text based version of a similar model may provide.

Certainly there is also a vast use of rich symbolic systems e.g. pictorial representation in a number of conventional systems. But there is a major difference between the pictorial representation of the EM approach and that of the conventional approach. Pictorial representation in conventional systems is an indirect counterpart of the interactions between human beings and the computer systems, while the pictorial representation of the EM approach offers a direct and immediate correspondence to these interactions. That is, the display of EM models represents a direct relationship to a state of the internal model within a computer system, whereas the display of conventional approach models is not a direct representation of the state of the internal model of a computer system.

EM as a generalisation of the spreadsheet

In 1986, S. J. Kaplan, principal technologist at Lotus, writing in the in-house journal expresses a broad vision for future spreadsheet use, *"Spreadsheets are the first effective environment for writing and running declarative programs. Naturally, the spreadsheet has become the tool of choice for problems that are best approached declaratively, whether they are in business, engineering, or other areas ..."* [Kaplan86, p. 138]. An EM approach embraces significant principles of the spreadsheet in constructing systems which are driven by a combination of human knowledge and automatic machine operation. For example, a spreadsheet environment allows the users to construct freely a working model of their operations based on their experience and then the machine is responsible for automatically retaining the relationships of each formula or function while processing the inputs. Human knowledge is represented by the choice of variables and choice of formulae. Automatic machine operation is achieved by dependency maintenance that will automatically handle the inputs and proceed to generate outputs. This kind of environment frees users from the worry of when and what proper machine instructions need to be given. So that users can focus more on the real work: the construction of the business models, for example.

Computer-based support systems built by the EM approach embrace spreadsheet principles in the sense that they represent personal knowledge of particular modellers or users, in addition to public knowledge in the form of propositional knowledge. An example of how a spreadsheet system represents knowledge of a particular user is as illustrated here. That is, an account system using spreadsheets created by an accountant of one company will never be identical to another one in another company of the same type. This is because there are no two persons with identical experience that could lead to identical knowledge even for the same practice. For example, a sales price plan for a company in which a sales representative has knowledge that the sales price is derived from the product fixed cost and overhead cost, is of less commercial benefit, compared with a sales price plan of another company that exhibits a deeper understanding that the overhead cost can be broken down into variable cost and contribution margin. So that a sales representative of the latter company with this kind of

knowledge delivered from an accountant who set this price scheme may offer a discount to customers in order to gain a bigger sales volume by agreeing to receive less contribution margin which means that he or she will receive less commission. However, with a bigger sales volume he or she might eventually gain more commission in total. This example suggests that, individuality and particularity that allows such components to be explicitly demonstrated, often lead to new insights in the same way as scientific experiments can lead to new discoveries.

Companies using the same accounting package will be forced to follow strictly each detail of the accounting practice provided by such software. That is to say, there is little room for human faculties to elaborate, and then to expand, their knowledge of their specialised practices. It is often the case that the design criteria chosen to cover most types of business, provide a comprehensive range of transactions that may not suit a particular company. For example, a company that does its business on a cash only basis, may not want a feature of credit transactions. However, this limitation reflects two sides of the same coin. On the one hand, it allows consistency of the system, especially when this involves many users of the system. On the other hand, it limits the contribution of humans to fit in with the calculating power of the machine. This may not always be the best organisation.

For the last two decades, computerized business decision modellers, have enjoyed the powerful end-user packages of spreadsheet systems for modelling purposes. Accounting, financing, sales forecasting, logistic and production planning, statistical analysis and many other similar business activities have been conducted, with the use of spreadsheet systems, in a way which is more understandable, more convenient, more accurate and faster than the traditional practice using manual methods and paper forms. Spreadsheet systems are obviously more comprehensible by general users and much less expensive compared with DSS developed by specialists. Nevertheless, the relative efficiency and the effectiveness of these two systems can be questioned. It is likely that it will never be comparable because they are often built on different scales. However, a practitioner in the oil refinery industry (a personal contact of the author) has stated that he usually works with spreadsheets to run any deci-

sion model, even if there is a large scale DSS operating in the firm he is working with. He said that the spreadsheet is much more convenient, and gives faster feedback, than if he were to work with the large scale system. He agrees that calculation results from spreadsheet models are less precise (this is because of the limited accuracy allowed by a spreadsheet system), but they are good enough to work on.

The reason why spreadsheets facilitate only semi or well-structured problems is because of the limitations of a system structure which serves only mathematical and logical arguments of only a few types of variables. This kind of system structure may be a result of the inherent concepts from the early days of computing era, as discussed in chapter 2, that is to focus on utilising computer capacity. It is commonly known that mathematical and logical arguments could only work well with a problem that is clearly identified in terms of its type of solution. Robson and Pemberton [Robson+95] stated that *"A structured design is a key element of spreadsheet modelling and one of the basic principles to be adhered to is the use of separate and self-contained areas of input, calculations and output ..."*; this emphasizes the key feature of the spreadsheet and at the same time the key limitation of the spreadsheet. The following statements explain what is meant by structured design. In doing financial forecasting, for example, certain groups of variables:

1. the range of input; for example, production costs, administrative costs, sales revenue, other monthly income, cost of goods sold and salary expenses;
2. the range of expected output; for example, revenue after tax, cash flow and dividend earnings;
3. the constraints; for example, tax rate, interest rate, tariff rate and consumer price index,

have to be recognized in advance and constructed prior to the modelling processes. Despite the fact that some of these variables are fixed and controllable while the others are not, all have to be assumed or given and entered into the model. In addition, the clear structure of their relationships have to be explicitly represented by definitions or formulae.

The structured design property of spreadsheet systems resembles most traditional general programming. That is, they are primarily designed to prescribe situations, not to reflect situations. Their inability to reflect situations is because of their limited data types and limited kinds of interaction. With this property of traditional modelling practice, no solution is offered for some particular kinds of business problem unless the problem itself, or some of its parameters, can be clearly identified. Most business problems are more like social science problems than physical science ones, in that they cannot always be described by abstraction-based mathematics and logic. There is no universal set of factors adequate to describe all business activities. This indicates that programming principles, based mainly on how well behaviour can be prescribed are not suitable for explaining business phenomena. This kind of problem needs a different treatment from what is given by conventional computer-based systems.

The EM approach takes seriously the need for human involvement and customising in model construction in a similar way to spreadsheet use. EM offers a generalisation of spreadsheet principles in terms of: variety of data types; access by multiple users, graphic-based visualisation; graphical-user-interface to access models which support EM's kind of interaction; possibility of using its models to operate or study a domain on a broader scale.

A computer-based support system as an instrument

EM models are developed from personal experience. The interfaces to the systems are considered, and constructed, from the very beginning of the development process. Throughout the whole development process, the interfaces are being shaped to correspond in the closest possible way to the users' experience of their counterparts in the real-world referent. These properties make human-computer interactions within the EM environment of a different quality from similar systems developed in conventional ways. This is because the systems are developed in a way that reflects the actual experience, or the real understanding, of the users of such systems. In some conventional methodologies, the interfaces to the systems are constructed almost at the end of the development process when most of the system structures have been con-

structed. This feature makes it difficult to have interfaces that really represent the users' experience. Having constructed the interfaces from the very beginning enables the systems being developed to possess the users' real requirements.

EM models can be considered as instruments to support human agents. An EM model is like an instrument in the sense that it supports the creation of a new task accomplishment. For example, a musician can play with an instrument to invent an absolutely new melody, while a carpenter may use a hammer to create a musical tune on some construction materials, but these two kinds of music composition are of different quality. One has a sound quality that can be appreciated by most people. The other is of radical quality and may not be appreciated by most people. However, the degree of achieving a new discovery depends on individual skills or experience of the domains. Like musical instruments, the performance of each piece of work largely depends on the musician's skill and experience, not the device on its own. On the other hand, computer-based support systems offered by conventional approaches, work more like tools. Tools used to accomplish a task must be used according to the way in which they were originally designed. Certainly, some tools might be arbitrarily used to achieve some other work for which they are not designed but such achievements are normally not significant. EM based models can be used more like instruments, and adapted to various uses as required. They can be regarded as an extension of 'cognitive models' or the 'mental models' of the modellers.

Chapter 4

Empirical Modelling Knowledge Representation

4.1 Introduction

The review and analysis of computer-based support systems in Chapter 2 indicates strongly that systems which can represent knowledge, or indeed re-present, or generate knowledge, are highly desirable especially for support systems. That chapter also drew attention to the limitations of current knowledge representation in computing. In response to this finding, this chapter investigates and identifies properties of an EM environment that make it possible to provide a special form of knowledge representation (KR). This experience-based form of KR provided by an EM environment is not entirely new, even in computer systems, because it has existed to a limited extent in spreadsheets. But it corresponds better to the needs of the business environment than others. This claim is based on our hypothesis that KR in an EM sense is more flexible and integrates different kinds of knowledge in a way which is closer to how people make sense of the world or solve everyday problems. Such sense-making and problem solving are the human capabilities central to business activities.

The chapter begins with perspectives on the terms 'knowledge' and 'representation' from different disciplines. The purpose of this is to show the relationships between those perspectives and the EM outlook. The knowledge representation that is possible in an EM environment is explained in terms of how it arises, what are its characteristics and how it can help provide a better computer-based support system. In a final section, the implications of such philosophical concepts for practical work and the development of useful modelling within the EM environment are discussed and illustrated.

4.2 Views of Knowledge and Representation

There are many ways of knowing. For example, processes of pure thought or cognition allow for the development of intellectual knowledge. Repeated experience of physical materials or artefacts give rise to skills or practical knowledge. By experiencing the emotions, for example, fear, anger, love, joy, feeling or affective knowledge can be developed. And spiritual experience can give rise to a different kind of knowledge. The end result of knowing is different kinds of knowledge obtained as a result of different kinds of experience, that is, intellectual experience, practical experience, emotional experience or spiritual experience. This leads to a number of theories in different areas of interest about this most difficult term 'knowledge' that try, for example, to define the scope and meaning of knowledge, to explain its structure or to illustrate the means by which it can be represented. However, there is no single theory that can claim to be a complete and universal explanation of it as yet.

This thesis makes no attempt to find a solution to the controversial debates about the theories of human knowledge. For example, the debates among the students of both Locke's and Berkeley's theory of human knowledge. Those theories, as described by G. J. Warnock [Warnock62, p. 36], are the origin of the twentieth century contemporary 'Causal Theory of Perception' and 'Phenomenology' respectively. This thesis also makes no attempt to settle the use of the term 'knowledge representation' as a universally agreed term. There is a wide range in the meaning given to 'knowledge' itself from the use in philosophy as 'justified, true belief', the everyday, or commonsense, usage through to the AI sense of *"background information or beliefs, which may, or may not, accurately reflect the external world"* [Way94, p. 64].

Knowledge is understood here as derived from, or gained through 'pure thought' and 'experience'. The former source is emphasised in rationalism and the latter is emphasised in empiricism. Both sources seem to be essential to constitute human knowledge. This viewpoint supports the idea that an EM environment that allows both experience-based (empiricism) and abstraction-based (rationalism) knowledge can complement a conventional computing environment (which is predominantly

abstraction based). An EM approach proposes that integration of kinds of knowledge, i.e. propositional, tacit and experiential kinds of knowledge, in the same environment will assist in strengthening the user's cognitive processes. Such an integrated forum for knowledge and beliefs is present in EM models or cognitive artefacts while users are interacting with them. These cognitive artefacts of EM are used as media to communicate the users' or modellers' knowledge of the area under study, either to themselves or to others. This leads to the suggestion that the EM approach can be used as an instrument to develop computer-based representations of the knowledge needed to understand better a situation, or to solve a problem.

EM is an approach to computer-based modelling which brings together propositional knowledge and human understanding of situations (experiential knowledge) in a system of two-way exchange. It is the case that human sense-making activities are brought within the scope of the data manipulation activity that allows experiential knowledge to be incorporated. What is meant by 'human sense-making activities are brought within the scope of the data manipulation activity' is that human-mediated meanings and interpretations are intimately involved with the computer processing of data and information. This close involvement of semantics (human processing) with electronics (computer processing) arises through the rich form of interaction typical of an EM environment.

The incorporation of experiential knowledge with propositional knowledge gives rise to contexts much richer than those which abstract computer states or operations can attain. Conventional computer operations represent states or situations in only an indirect way so that the human interpreter has to exercise great skill and imagination to connect them with experience. For example, when a programmer has to deal with removing bugs in conventional programming computer states, and state transitions may not be readily comprehended and may not be readily applicable to the solution of everyday problems. They are abstract and, comparatively speaking, inflexible. EM constructs and uses computer systems as cognitive artefacts. The use of computer systems as cognitive artefacts allows them to represent a range of knowledge wider than the propositional and to be consistent with the growing knowledge of the modeller. In

this way designers or users of such computer models can more easily make use of their knowledge; they inevitably modify the models as they use them to meet their own objectives. EM goes beyond the conventional use of models bringing them into service as instruments in problem solving. It can be compared with experimenting and testing, as this is understood by scientists, until a set of objectives is achieved.

4.2.1 Philosophical view

Bertrand Russell [Russell79, p.23-24], a well-known philosopher, argues that there are two sorts of knowledge: **knowledge of truths** and **knowledge of things.** Knowledge of truths refers to, *"... the sort of knowledge which is opposed to error, the sense in which what we know is true, the sense which applies to our beliefs and convictions, i.e. to what are called judgements ..."* or, in other words, when we, *"know that something is the case"*. Knowledge of things refers to (a) **knowledge by acquaintance** which occurs when we experience sense-data and is, *"simpler than any knowledge of truths, and logically independent of knowledge of truths"*, and (b) **knowledge by description** which occurs in cases where we have true judgement without acquaintance and, *"always involves ... some knowledge of truths as its source and ground ..."*. These descriptions of knowledge by Russell seem to be broadly accepted and are often repeated in the literature.

L. A. Reid [Reid61], a professor of philosophy of education, writing in the early 1960's states that the use of the term 'knowledge' by both philosophers and scientists is often limited to, *"that which can be expressed in propositional statements which are 'true', which affirm 'something which is the case' to be so ..."*. This might be the result of the influence of science. Such influence leads to an emphasis on formal structures in the scientific approach, in which evidence is provided by publicly agreed procedures or rationally established facts, in order that any field of study can be granted credibility. About forty years later such limitations still seem to be commonly assumed as shown in Trigg [Trigg99] who is still debating the principles of knowledge in the same direction as Reid. The accidental truth of real-world phenomena, which is derived from knowledge as we experience it, cannot be effectively explained by the necessary truths of mathematics which underlie the propositional knowledge of science. Trigg puts it

in this way, *"The necessary truth of mathematics cannot be assumed to reflect the contingent truths of the physical world ..."* [Trigg99, p. 194]. His discussion supports the idea that, *"Research into the character of natural irregularity has demonstrated that what science can do depends on the nature of the reality it confronts. Science can produce a complete set of deterministic laws only if physical reality is itself deterministic ..."* [Trigg99, p. 193]. These debates seem to reveal the main idea that mathematical expressions, which are assumed to be a major means of representing scientific theories or knowledge of truths, cannot be used as the only and absolute means of describing every single phenomenon. To be able to understand a real situation, here and now, is to be able to perceive and describe things as they are. Trigg states, *"... it still remains the case that any claim to knowledge must be grounded in the way things are ..."* [Trigg99, p. 1].

Perhaps developments in terms of the dramatic increase in material wealth and in technological advancement that have been brought about by the introduction of scientific approaches and their subsequent influences, e.g. in respect of the domination of formal analysis, quantitative measurement and the proof of efficiency and effectiveness judged only in terms of measurable results, blur the fundamental facts. Reid points out how unrepresentative a narrow scientific perspective on knowledge is compared with the more general use of the term 'knowledge',

> *Yet the range of general use is much wider and richer. We say we 'know' in sense perception (long before the use of the words), and through feeling; we 'know' a poem or a figure; we 'know' other persons – acquaintances, friends, lovers; we 'know' good and bad, right and wrong; we may even claim to have some sort of 'knowledge' of God ...*
>
> [Reid61]

What Reid refers to here are kinds of inarticulate knowledge which have been researched and discussed in a number of works in different disciplines [Winograd+86, Baets98]. Awareness of these kinds of knowledge is important for Empirical Modelling because they give rise to significant observables for human agents. Patterns of interaction with such observables may allow integrated kinds of knowledge, includ-

ing inarticulate knowledge, to be presented while using or developing an EM computer-based system. For the sake of sharing or communicating this knowledge in EM one experience (of a model, or computer artefact) can be a representation of a similar experience of the real world referent.

Reid also points out that the use of words in sentences, or those mathematical statements that form symbolic representations, could never completely describe what a person may know. He argues for the inclusion of **signs** and **symbols** within the representation of human knowledge and experience. At this point, a spreadsheet system is a good example of how the inclusion of signs and symbols in representing human knowledge can be achieved, to a certain degree of satisfaction, within a computer-based model. The computer-based model here shows a form of human knowledge representation. This is owing to the nature of a spreadsheet system that allows new observations to be included, instantly and endlessly, into an existing model. Such new observations are made as a result of humans seeing their significance. Therefore the ease of including new observations made by humans into a computer-based model at any stage in a spreadsheet system, supports the inclusion of signs into such a model which is the representation of human knowledge. That is to say, the significance of not neglecting human involvement in the computer-based process, is to give greater support to human knowledge as it is actually encountered in experience. Reid also argues that only the mind can possess knowledge and that the mind can become 'lit up' with knowledge through the efforts and experience of expressing oneself. Thus he writes:

> *The knowledge expressed in statements which are 'true' uses symbols of an appropriate kind, very often words in sentences, sometimes mathematical statements. But in every kind of knowledge and experience – sense-perception, science, art, myth, religion, personal encounter – symbols and signs are involved, each overlapping field having a kind of 'logos' of its own ... On the one hand, knowledge is never identical with its symbolic articulation – as often seems to be assumed. And no summation of statements, however complete, ever adds up to knowledge. Knowledge is possessed only*

> *by the mind, which becomes illuminated through its different efforts and experience of articulation. The mind is able to re-experience with more discriminating insight because of these efforts and experiences ...*
>
> [Reid61]

Michael Polanyi [Polanyi67] shifted his interest from technical scientific subjects to philosophy when he realised that the theory of physical sciences cannot explain the nature of things. A very straight-forward example is that physical scientists cannot yet explain why they have to structure scientific subjects as they do. His work in philosophy emphasised 'tacit knowing' as a powerful fundamental source of knowledge. He writes,

> *... tacit knowledge dwells on our awareness of particulars while bearing on an entity which the particulars jointly constitute. In order to share this indwelling, the pupil must presume that a teaching which appears meaningless to start with has in fact a meaning which can be discovered by hitting on the same kind of indwelling as the teacher is practising. Such an effort is based on accepting the teacher's authority.*
>
> *Think of the amazing deployment of the infant mind. It is spurred by a blaze of confidence, surmising the hidden meaning of speech and adult behavior. This is how it grasps their meaning. And each new step can be achieved only by entrusting oneself to this extent to a teacher or leader. St. Augustine observed this, when he taught: "Unless you believe, you shall not understand" ...*
>
> [Polanyi67, p. 61]

He points out that much scientific discovery is based on beliefs which are directly associated with tacit knowing. Polanyi describes a number of examples to show that people know more than they can tell, which is what is meant by tacit knowing. One example is that we usually cannot tell how we recognize a face we know. But in a method introduced by police authorities a large collection of pictures showing a variety of facial features i.e. eyes, noses, mouths and ears, are provided, and from these

the witness selects the particulars of the face he knows, and the pieces can then be put together to form a reasonably good likeness of the face. This suggests to Polanyi that this kind of knowledge can be communicated when adequate means for self expression is given [Polanyi67, pp. 4-5]. He points out that the successful act of communication displays this kind of knowledge. He explains the structure of tacit knowing as the following:

> *... by working out the structure of tacit knowing. This structure shows that all thought contains components of which we are subsidiarily aware in the focal content of our thinking, and that all thought dwells in its subsidiaries, as if they were parts of our body ...*
>
> [Polanyi67, p. x-xi]

Polanyi refers to the demonstration of Gestalt psychology that, *"... we may know a physiognomy by integrating our awareness of its particulars without being able to identify these particulars ..."* as closely linked to his analysis of knowledge. This demonstration, Polanyi believes, is a response to the Gestalt theory that, *"... the perception of a physiognomy takes place through the spontaneous equilibration of its particulars impressed on the retina or on the brain ..."* [Polanyi67, p. 6]. He argues that the perception on which Gestalt psychology is centred is the poorest form of tacit knowing. He argues for, *"the outcome of an active shaping or integration of experience performed in the pursuit of knowledge"* as opposed to Gestalt perception theory. He suggests that the shaping and integrating is, *"the great and indispensable tacit power by which all knowledge is discovered and, once discovered, is held to be true"* [Polanyi67, p. 6].

'Knowing' according to Polanyi refers to both 'knowing what' and 'knowing how' or in other words 'intellectual' and 'practical' knowledge. He argues that, *"these two aspects of knowing have a similar structure and neither is ever present without the other"* [Polanyi67, p. 7]. He gives an example for this claim that, *"... the art of diagnosing intimately combines skilful testing with expert observation ..."* [Polanyi67, p. 7]. He points out that, *"The scientific outlook appeared to have produced a mechanical conception of man and history in which there was no place for science itself. This conception denied altogether any*

intrinsic power of thought and thus denied also any grounds for claiming freedom of thought ..." [Polanyi67, p. 7].

In summary, discussion in this section shows that among philosophers' views of human knowledge, those of Russell, Reid, Trigg and Polanyi, come to the similar conclusion that there are two major kinds of knowledge: knowledge that can be articulated and knowledge that cannot be directly articulated. Human knowledge is always a combination of the articulate and the inarticulate. In addition most of them argue that scientific beliefs that value only articulated knowledge, i.e. mathematical and logical explanations of phenomena, distort the existence and the significance of inarticulate kind of knowledge which actually is the basic ground of developing articulate knowledge. Almost any development of scientific theory begins from inarticulate knowledge. That is, usually, scientists with years of experience in the field may know that something can be the case but they can't explain very well yet how and why. Thus they have repeatedly to set up experiments to test what they have in mind, that is, their inarticulate knowledge, to make sure that what they know is really the case. Only then can an improved scientific theory emerge. In the same manner of scientific experiments, EM models were developed by exercising both articulate and inarticulate knowledge of either the modeller or user.

4.2.2 Psychological view

The psychologist's view of knowledge and representation is mainly focused on demonstrating how knowledge is represented in the human mind. A major analogy used to illustrate how the mind might work is the comparison of mental processes to computer processes. For example, P. Johnson-Laird [Johnson-Laird93] holds that the human mind works like a program whereas the brain works like computer hardware.

There is a large body of literature by psychologists in regard to knowledge and representation, but not very much having the same orientation for the term 'knowledge' as it is used here in this thesis. A work that does have a similar outlook is the work of Rumelhart and Norman [Rumelhart+88]. They define the term 'representa-

tion' as follows *"... a representation is something that stands for something else. In other words, it is a kind of model of the thing it represents. We have to distinguish between a representing world and a represented world. The representing world must somehow mirror some aspects of the represented world ..."* [Rumelhart+88, p. 513]. They suggest that, *"... the same characteristic in the represented world can be represented very differently in different representing worlds ..."*. They argue that the most important point of a representation is that *" ... it allows us to reach conclusions about the thing being represented by looking only at the representing world ..."*. They point out that their theories of representation are actually representations of a representation. By this they refer to representations of mental activity that are in turn a representation of the environment. They argue that representational systems include both representation and process, of equal significance. Process refers to the practice of evaluating and interpreting representations.

The relationship between knowledge, representations and the process of developing representation of our understanding of the world, as perceived by Rumelhart and Norman as psychologists, corresponds quite closely to what is proposed in the EM approach for computer-based support environments. That is, both the representation and the process of representation (the evaluating and interpreting of models to represent the real world referent) are equally significant factors in reflecting the real world situation. Because of this it is reasonable to expect a faithful model of the real world referent.

Three basic families of representational system as described in [Rumelhart+88]. These are:

1. **Propositionally Based Representational Systems** – In this kind of system, the concepts of the world are represented by formal statements, that is a set of discrete symbols, or propositions, is assumed to represent knowledge. Schemes of propositional representation are: Semantic features, Semantic Networks, Schemas and Frames. In general, they are designed to represent meanings and to represent information stored in long term memory.

A semantic feature approach focuses on the representation of word meanings by lists of features or attributes. The assumption made by this approach is that

concepts are properly represented as a set of semantic features or attributes. Within this approach, it is assumed that in all models conceptual knowledge is represented by a set of features, and that these features include all attributes characteristic of the concept being represented. Semantic feature models are claimed to provide good accounts of a wide body of data. However, most work has been done with simple concepts and it is not clear how such models would represent complex facts.

Semantic networks include in their representational systems both lexical and sentential knowledge. In this representational system, knowledge is believed to be represented by *"... a kind of directed, labelled graph structure in which the basic structural element is a set of nodes interrelated by relations ..."* [Rumelhart+88, p.523]. However, both semantic features and semantic networks share the common characteristic of representing all knowledge in a single and uniform format.

Schemas and Frames focus on a higher level of structure: supra-sentential knowledge. Their goal is to change the level of discourse, not to remedy the expressive problems of the earlier two approaches.

2. **Analogical Representational Systems** – In this kind of system, the features of the world which are continuous are represented by direct correspondence between the represented world and the representing world. Continuous variables are traditionally assumed to be needed to represent knowledge. Rumelhart and Norman refer to researchers in this area, for example, Shepard, Cooper and Kosslyn, propose that, *"... the knowledge underlying images is analogical rather than propositional ..."* [Rumelhart+88, p. 546]. The conclusion made from the experiments of the analogical representational system is that, *"... people can create images that are surprisingly veridical and that can be processed in the way that an actual picture would be processed. Imagined objects are certainly analogs of the physical objects which they represent ... , however, the matrix representational format is probably not sufficiently general for use in many cases in which we use our imagination to solve problems. It seems likely that a richer representational format is necessary ... "* [Rumelhart+88, p.554]

3. **Procedural representational systems** – In this kind of system, the concepts of the world are represented in terms of procedures. The procedure which is used in the actual performance of the skill, and not simply the ability to do the task, is considered to represent knowledge. To distinguish the actual performance of the skill from the ability to do the task may be justified by the fact that one performer may be more efficient than another at the same task. In addition, an action system is used to interpret directly that particular form of representation.

A summary of the psychological view of knowledge and representation is that most actual representational systems of knowledge in the human mind are hybrids of these three main representational systems of knowledge: propositional, analogical and procedural. This means that any system that aims to represent how a human mind works (the *represented world* in the terms of Rumelhart and Norman) needs to support a hybrid representational system of all three kinds mentioned above.

4.2.3 Business operational view

Baets [Baets98] states that in managerial sciences there is relatively little concern for defining managerial knowledge compared to that in the cognitive sciences. He gives an example of a definition given by Kim's work in 1993 that, *"... knowledge is a combination of "know-how" and "know-why ..."*. However, he does focus on works of Firebaugh in 1989 and Nonaka et al in 1994 on the two kinds of knowledge: explicit knowledge and implicit knowledge, and links their hypotheses with his ideas as described below. These two kinds of knowledge defined by Firebaugh and Nonaka et al seem to be consistent with the general distinction in kinds of knowledge which has been noted already: articulate knowledge and inarticulate knowledge.

Explicit knowledge is transmittable in formal and systematic language and is dealt with in knowledge-based systems. Implicit knowledge or tacit knowledge is, as suggested by Firebaugh, logically entailed in the system and is, as suggested by Nonaka et al, deeply rooted in action, commitment and involvement in a specific context. Implicit or tacit knowledge has a personalised quality, involves cognitive and techni-

cal elements and also involves context. Baets points out that, in the management context, tacit knowledge is the kind of knowledge that deserves most attention. This is because " ... *tacit knowledge gets used in managerial tasks and tacit knowledge is the knowledge which makes the difference. Key to acquiring tacit knowledge is experience. ... It proves extremely difficult to extract this kind of "knowledge" ... The capacity of an organisation to take effective action is based on tacit corporate knowledge ...*" [Baets98, p. 54]. Baets suggests that tacit knowledge is acquired through experience. He points out that it is extremely difficult to extract tacit knowledge. It is not because a person does not want to share it, but because it is difficult to make it explicit. Baets suggests that to deal with tacit knowledge, a number of cognitive elements are involved.

He refers to 'mental models' as defined by Johnson-Laird in 1983 as the central cognitive elements for managerial problems. Mental models, as defined by Johnson-Laird are working models of the world, formed by human beings, by creating and manipulating analogies in their mind. Baets also refers to the description given by Senge in1990 of mental models, "*... as deeply held internal images of how the world works which have a powerful influence on what we do because they also affect what we see ...*". Baets refers to Kim's description in 1993 that, "*Mental models provide the context in which to view and interpret new material and they determine how stored information is relevant to a given situation ...*". He himself suggests that, "*... mental models represent a person's view of the world, including explicit and implicit understanding ...*" . He argues that it is not the reality that matters, but the perception of that reality. He links the perception of reality to the significance of the context of learning and knowledge. He argues that individual intelligence can hardly develop knowledge that goes beyond the capacity of the individual, in a system of individual elements. Such knowledge is not stored, it is created each time that there is an interaction of the different elements [Baets98, p. 33].

The work of Baets as described above shares and supports arguments made in this thesis on the significance of mental models and inarticulate knowledge, especially for performing business activities. It also explains the way in which the mind incorporates both explicit and implicit knowledge and the significance of a system that can support the growing of knowledge instead of merely the storing of knowledge.

4.2.4 Computational view

In the computational view, computer scientists try to represent different kinds of knowledge, e.g. knowledge in an accounting process, in disease diagnosis and in aviation navigation by different kinds of representational systems that are supposed to suit well the individual needs of each domain. As mentioned in the earlier two sections, the scope of the terms 'knowledge' and 'representation', even in the same area, is so broad that again a discussion in this section will only present a general view that is related to the theme of this thesis.

Knowledge representation has been extensively discussed and debated in the area of Artificial Intelligence. The belief of AI researchers is described by a statement of E. C. Way [Way94] that, *"... knowledge seems to be a prerequisite for any kind of intelligent activity ..."*. So people in AI are focusing their researches on ways of representing different kinds of knowledge in computer systems. Examples of kinds of knowledge and ways of representing them in AI are comprehensively described by the following statements of Way.

> *All areas of research in AI from game playing to expert systems, from computer vision to natural language processing, require vast amounts of knowledge about the domain within which the system will be operating. Chess programs need to know not only which moves are legal for each piece but also all kinds of heuristics or rules of thumb for deciding the best strategy, and for knowing when the game is, for all intents and purposes, lost or won. Expert systems, of course, are the very embodiment of an expert's knowledge and experience translated into a series of condition-action rules (when this condition occurs, then take that action). In computer vision it was found that there is too wide a gap between raw image data and any kind of intelligent use of what is 'seen'. In order to make sense of these images various kind of knowledge are necessary; for example, knowledge about brightness and brightness change, knowledge about the relation of these changes to texture, edges and surfaces, etc. Natural language processing also requires vast amounts of*

> *knowledge; knowledge about the syntax of language, the meaning of words, knowledge about what is assumed as well-known in a conversation and what is implied by a particular choice of words ...*
>
> [Way94, pp. 61-62]

Such domain knowledge as discussed by Way above is huge and complex. If it were able to be completely derived and stored in any computer system, it would be very expensive and might not be worth the effort in terms of usage. However, there are many difficulties for AI in representing knowledge. Those which have been mainly considered in this research are: firstly, how to store the enormous amount of common-sense background knowledge which a person may have [Way94]; secondly, how to represent or incorporate inarticulate knowledge (tacit knowledge, skills knowledge or experiential knowledge); thirdly, how to build a computer system that can behave intelligently.

The classic application of computing to knowledge representation is in the construction of expert systems. This has been discussed briefly in § 2.4.2. The complicated and unknown nature of the mind leads to difficulties in the AI area, especially for those who wish to develop a real expert system [Searle92, Crane95, Way94]. The difficulties arise because:

1. people may not realise what they know and, in this case, it is hard to obtain complete expert knowledge to be stored in a computer-based system;

2. people have a special capability to learn and adapt what they know to the situation at hand although they have never encountered it before. In this case new knowledge is gained through open-ended interaction with the environment. That is, people are able to develop new kind of knowledge while a computer-based expert system may develop only a new knowledge of the same kind;

3. people's judgement is faster than computer operations. This is owing to their ability of discrimination among choices. Such discrimination also includes cases where human concern is a priority, where more realistic choices are made and where economy of scale operates. This refers to the comparison

that is based on both a variety and a number of analytical choices to be made. People are relatively good at discriminating between necessary and unnecessary factors that need to be taken into account. It is difficult to give computer systems any degree of discrimination; typically they either search exhaustively, or as directed by some pre-defined criteria.

4.3 Knowledge Representation in EM

A book is a form of knowledge representation in the sense that it represents an author's *domain knowledge*. However, knowledge that is represented in any book is articulate knowledge or descriptive knowledge as defined by Russell. This kind of knowledge can be conveyed from its origin (a writer), through a kind of **one-way interaction** (reading) via a medium (a book), to the knowledge recipient (a reader). Reading is a kind of interaction that can evoke intellectual knowledge or cognition, as long as the readers can relate the topic discussed by the author to their own experience or are already familiar with the topic. While reading books, readers are manipulating messages or data internally. They utilise their experience of the subject area or the related areas. Their construals or interpretations of each message are guided by their experience, so that two readers of the same book may often acquire different messages.

A conventional computer system provides a similar kind of knowledge representation to a book. The major difference is the availability of *forms of representation* and *means* of access to manipulate the represented knowledge. That is, the users can externally intervene by giving input and requesting a certain way to manipulate data. This process is similar to the process that occurs when readers interact with their thought processes or imagination while reading books. However, the computer system allows an explicit form of evoking intellectual knowledge. The opportunity to visualise the result, or sometimes, both the process and the results of the manipulation by sight (not the mind) makes a strong impression. The evidence for this is that, in an educational environment, the most recommended learning aids are visual aids. This is on account of the fact that about 80% of the total information taken in by learn-

ers is through sight as compared with other senses such as hearing (7%). The interaction allowed in a computer system is two-way interaction. That is, for example, the users put in some inputs or queries, then the computer may respond with a result, or an answer, or a variety of choices of results and answers. The availability of a variety of results or answers in a responsive manner often evokes further thought in a wide-ranging way. The two way interaction provided by computer systems is like an interaction of two people in a conversation. This is the case only if the computer system is very 'open'. A conversation involving two or more people often gives rise to a situation in which one person's response evokes another person's cognitive process.

A conventional computer system provides the facility to exhibit or articulate the cognitive process. However, the scope is limited because the interactions available are pre-conceived and fixed. This means that the processes of evaluating and interpreting the representations are limited and, according to Rumelhart and Norman, this is crucial: *"The processes that evaluate and interpret the representations are as important as the representations themselves ..."* [Rumelhart+88, p. 516].

An EM environment offers a different kind of knowledge representation from that of books or conventional computer-based systems. The significance of the EM environment is that human sense-making activities interact closely with the data manipulation activity. Therefore the significance and flexibility of human interpretation, which is mainly based on common sense or a priori knowledge and experience, are not lost, in contrast with systems that allow the manipulation of symbols only by rules stored in the machine. In reading books, human subjects are included in the processes of manipulating the data, but the lack of facility to exhibit their thought processes externally prohibits them recording (and possibly even realising) much of what they are doing in their thought processes or their knowing. Familiar tools such as paper and pen are artefacts to assist people in their thought process or cognition. They support the cognitive process by allowing users to illustrate visually and freely what they know, in the way they understand it. That is to say, people need cognitive aids to support and, probably, extend their cognition.

In the EM environment, human agents are able to build models which are like paper and pen on a grand scale: a new modelling medium allowing, in principles, the immediacy of mental models and similar feedback as from a physical model. This allows the user to control and monitor the manipulation and interpretation of data and so also to exhibit their thought processes and cognition. This supports the claim that there is real knowledge representation of an integrated kind in an EM environment. This is owing to the fact that the modeller is regarded as part of the EM environment and only humans are capable of knowing and thus only humans can possess knowledge. (See the end of the quotation on p.122-123.)

The two kinds of knowledge that are represented in an EM environment

The two kinds of knowledge defined by Russell are *knowledge by acquaintance* and *knowledge by description;* an EM environment enables the representation of these two kinds of knowledge and utilises the significance of each kind. Firstly, ***knowledge by acquaintance*** is included in an EM approach through the concept of observables. Russell refers to this kind of knowledge as that of which there is direct awareness without the intermediary of any process of interference or any knowledge of truths. The concept of observables in EM is like Russell's idea of sensory data which makes up the appearance of things with which a person has acquaintance, things immediately known to a person as perceived. This line of thought is opposite to the concept of variables used by conventional computer systems where they are the consequence of mathematical abstractions. The significance of knowledge by acquaintance is that it can be incorporated into new knowledge. Secondly, ***knowledge by description*** can be related to those formal structures of mathematics and logical statements which can also be provided within an EM environment. Russell states the significance of this kind of knowledge:

> *The chief importance of knowledge by description is that it enables us to pass beyond the limits of our private experience. In spite of the*

> *fact that we can only know truths which are wholly composed of terms which we have experienced in acquaintance, we can yet have knowledge by description of things which we have never experienced. In view of the very narrow range of our immediate experience, this result is vital ...*
>
> [Russell79]

Knowledge by description, or propositional knowledge, cannot simply be equated with the knowledge represented in definitions. The way different forms of knowledge are represented in EM (or languages generally) is complicated. For example, a simple definition of the form

```
lamp_position = table_position + offset
```

does represent propositional knowledge about the position of the lamp. But it is more than this because it represents a persistent one-way relationship as well as a current situation. This relationship knowledge might be experiential knowledge ("I have always observed this.") or the prediction of a theory ("In the present of gravity, and suitable friction, this is what will happen.") or it might be a matter of design (consider the dependencies representing the speedometer markings in § 3.2.2). It can be seen that this is a one-way relationship from the fact that a definition is non-symmetric. For example, Newton's second law is often expressed in the equation

```
Force = mass x acceleration
```

where each quantity has an equal right to be made the subject of the formula. In contrast, a definition is a one-way constraint. For example, in the vehicle cruise control system (see [EMWeb]) Newton's law appears in the following form

```
accel = (tracF - brakF - gradF - rollF - windF) / mass
```

with the meaning that as the throttle is opened and the tractive force due to the engine is increased then the acceleration will automatically increase. But not conversely. To calculate the force acting on a pilot due to the plane's acceleration a definition is needed of the form

```
force_on_pilot = mass x acceleration
```

A kind of interaction that evokes knowledge

Although Baets adopts a very different technology – that of neural networks – some of his important motivations are shared by EM. Among hypotheses made by Baets, there are two that provide support for similar concepts adopted in an EM approach. The first hypothesis is that mental models consist of both explicit and implicit understanding [Baets98, p. 54-55]. The second hypothesis is that the knowledge creation process involves parallel activities. Baets illustrates this nature of knowledge creation by what is going on in a company meeting to get a consensus or a decision. He states, *"When one person brings in an idea, another person may get yet another idea. Ideas build on each other and while one person is speaking, others are thinking ..."* [Baets98, p. 66]. According to Baets, this is a way that knowledge gets created, that is, as a parallel process. These two hypotheses are similar to ideas of concern in EM. Baets's first hypothesis that mental models consists of both explicit and implicit understanding, is in line with an EM concern that a model is an extension of people's mental models within which there is a combination of articulate and inarticulate understanding of the problem domain or the situation under study. The articulate kind of knowledge or understanding can be represented by formal statements of conventional computing. The inarticulate kind of knowledge or understanding can be known or understood through interactions allowed between the human agents and computer agents. Baets's second hypothesis is that the process of knowledge creation is a parallel process. This idea is in line with an EM approach that model-building proceeds in parallel with the evolving interpretation of model and associated new knowledge.

The parallel nature of how human brains work is demonstrated by the fact of life that people are often *'thrown into a situation*[1]*'* in such a way that they can only rely

1. this refers to the term 'thrownness' introduced by Heidegger and cited by Winograd and Flores [Winograd+86].

upon their previous experience to handle the situation. The process of referring back to previous experience usually makes use of a combination of a number of different kinds of experience. Thus, a parallel process of interactions with different sorts of experience is established. This situation of being 'thrown' in unexpected circumstances encourages people to interact deliberately – that is with full engagement of the mind – to be able to handle the situation well. This kind of deliberate interaction with the environment, guided by associated experience, often results in gaining new understanding or having new knowledge.

In the EM environment, there is not only a chance to utilise one's experience, but also a chance to explore new experiences which is a source of enhancing one's intellectual knowledge. This is possible because of the **two-way-exchange-interaction** allowed in the EM environment. This two-way-exchange-interaction can be compared with interactions like holding a conversation, driving a car or performing artistic works. The essence of this kind of interaction is that the open, immediate feedback in response to individual action, provokes a rapid and direct response in tune with that action. There is a phenomenon of stimulus-response that shapes these activities. The two-way-exchange-interaction is not the same as the interaction between a modeller or user and a conventionally-based computer model, in which the interaction seems to be a kind of fixed stimulus-response that would not lead much further in utilising one's knowledge or accruing new knowledge.

In contrast, reading is an example of one-way-interaction – that is, there is no response from the other party in the interaction. Nevertheless, this kind of one way interaction can lead to the use of one's knowledge and the possibility of gaining new knowledge. This is because, when people are involved in imaginative, creative and analytical thinking, they consult their experience to construe or interpret the things they are encountering or interacting with. However, the lack of an uncircumscribed response from the other party reduces the chance of one's knowledge being evoked deliberately. The interaction with a book or a conventional computer-based system as compared to the interaction allowed in the EM environment can be described by the proverb that, *"I hear and I forget, I read and I remember, I do and I understand"*. That is, in

strictly following every step of the computer interface instructions, the users of conventionally-based computers are working in an environment similar to that of listening to a teacher instructing them how to do things, or reading instructions from books to solve a particular problem without their own initiative. They get access only to propositional knowledge or articulate knowledge. They tend to forget after a while if they are not involved. They may remember but because they don't really understand how it works, they seem not to be able to apply what they know to other similar problems. In an EM environment, users get access to propositional knowledge and, on top of that, the open-ended interaction allowed in an EM environment encourages users to learn things by doing. That is, the open-ended interaction of an EM environment supports users to initiate any interaction based on their knowledge. In this kind of environment, they are more inclined to reveal what they know. Then being able to interact arbitrarily with a situation or problem domain with what they know, they may gain insight. Insight allows people to apply their knowledge.

The contrast between one-way interaction and a two-way exchange interaction environment is illustrated in an EM model that has been developed in connection with the implementation of Noughts and Crosses (OXO) (see [Beynon+94 and EMWeb]. With reference to Figure 6, the first window ('Geometry') introduces the concept of lines on the grid. Each winning line is represented by a different coloured line. These are animated to more clearly depict the winning lines. The 'Status' window introduces the concept of pieces. Xs and Os can be placed on the board at random. There are no rules at this point. Counters can be overwritten or erased. Several identical counters can be placed in a row (i.e. you do not have to take turns in placing a counter). The 'Sqvals' window starts to introduce values for each move into the model. Each position on the board displays a value. This is a score for that square. These scores are calculated by considering each possible winning line which goes through that square. When considering a possible winning line, it looks at how many counters of each type are on the line already. It then gives that line a score. The window labelled 'Play' shows the scores the computer evaluated on its turn. The square it chooses is highlighted by a red square. The window labelled 'Gamestate' is the win-

dow you actually play in and it introduces the concept of having turns and abiding by the rules of play.

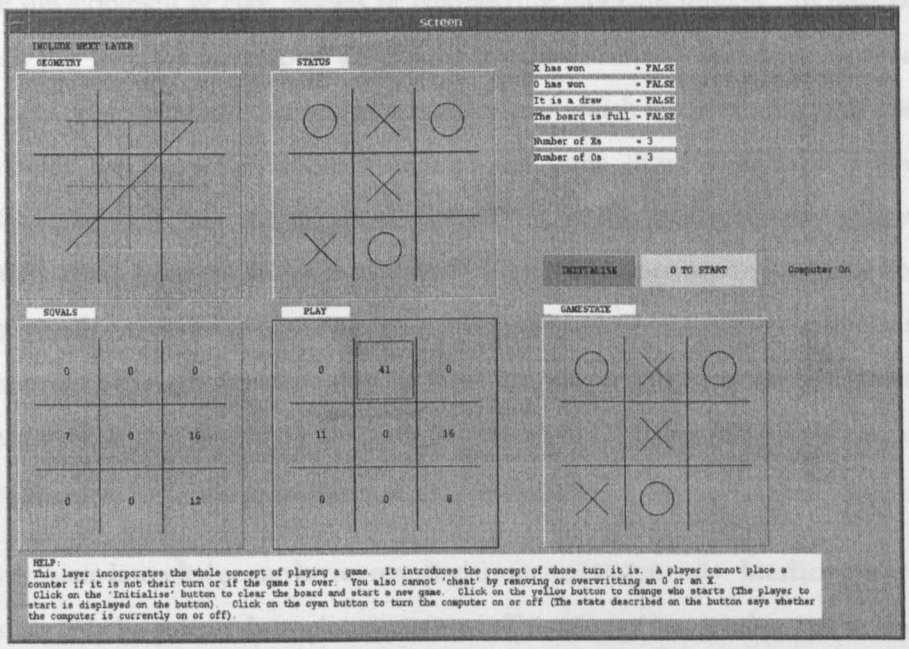


FIGURE 6. A screenshot of the EM version of OXO

A conventional version of OXO (say in Java) is an example of one-way-interaction, while the EM version represents two-way-exchange-interaction. They both represent knowledge of how to play the game. However, the kind of knowledge represented by those two systems is of a distinct quality. The significant difference in the quality of knowledge is based on the kind of interaction. That is, free and uncircumscibed interaction that allows users to experiment with the system like scientists can do in their laboratory experimentation, is provided in the EM environment. Through open-ended interaction, a person is allowed to freely experiment with a subject of interest. This very act of unbounded investigation raises opportunity for one to make incidental or radical discovery that adds to one's knowledge, that is to say, one gains new knowledge. The Java-based version in which the interaction with the system is circumscribed, can provide only the intended knowledge of how to play a game, as a book may provide to its readers.

An EM approach uses the term 'cognitive artefact' to describe a view of our use of computer systems, in accord with our concept of open-ended interaction. That is, these cognitive artefacts are constructed through modelling processes in a way similar to the development of clay models or engineering mock-up models. In developing models this way, the developers are invited to engage actively and freely in redefinitions (possibly via mouse actions or other devices) which represent physical interactions with the cognitive artefacts and, at the same time, to interact internally intellectually with their mental states. Interaction of this nature resembles how we normally come to know things in our daily life through interaction – whether objectively, or subjectively – with the environment, and through interaction with our mental models. Through developments of this kind, the cognitive artefacts that evolve represent both articulate and inarticulate knowledge of the real world situations under study, or of problem domains of the developers.

Articulate knowledge can be represented in formal statements. Inarticulate knowledge can only be gained through interactions. This kind of knowledge can be referred to as 'personal knowledge' which is hard to express or describe completely. People cannot articulate precisely all they know. This may be because what is known only becomes apparent in dynamic situations. It is difficult to describe comprehensively one's knowledge even for a single area of interest. It is interesting that we have the capability to apply and relate what we know to a new situation that we have never encountered before. Most of the time we use what we know through interactions with the world to learn and gain new knowledge. However, the level of this kind of capability varies from person to person and is partly based on the degree of interaction that an individual has with the world. The degree of interaction includes both the opportunity for, and the openness of, interactions. The greater the opportunities and openness of a particular model, the greater the possibility of increasing knowledge.

Experience-based knowledge through active response and adjustment

A kind of knowledge that is included in an EM environment apart from propositional knowledge is experiential knowledge. Experiential knowledge as referred to in this thesis is a kind of inarticulate knowledge. Inarticulate knowledge is a kind of knowledge where people cannot say or describe what they know. For example, a manager cannot easily tell what a market situation is like. But from her experience, she has experiential knowledge of different market situations and this knowledge influences her decision making or problem solving. This example gives two meanings of the term 'experience'. Firstly, it is used in the sense that a person who has been involved with a certain activity a large number of times and knows much about it is said to be 'experienced'. Secondly, it is used in a narrower, more specific, sense that while engaging in an activity a person is being aware, in the here and now, of the elements of a situation. It is in this second sense that experiential knowledge arises through *active response* and *adjustment* during interaction. The use of these two terms is borrowed from Reid to characterise a kind of knowledge that cannot be articulated, but is a kind of knowledge that is inseparable from the capacity to have experience: experiential knowledge. We use the term *active response* to mean the reaction given as a result of what a person is engaging with and the term *adjustment* as the shaping process to match the situation. Therefore, the term experiential knowledge used here in this thesis, refers to a form of people's knowledge that they use in reacting appropriately to each circumstance with which they engage. We believe the meaning given to these two terms by us shares the view of Reid.

> *Human articulation, at least in any mature form, is always in terms of symbols of one sort or another, and the study of symbols is part of the study of knowledge... But though articulate human knowledge uses symbols, it would be a great mistake to think that 'knowledge', at least of some kind, does not long precede human symbolic articulation. It is implicit in individual human experience. Though we cannot in our maturity refer to any instance of a piece of 'knowledge' without using symbols in expressing it, what is expressed*

> *symbolically does not comprise the whole of the knowledge. ... And in the history of life on the world, we must assume that the world becomes 'known' to the sub-human organism – and to the human organism before the stage of speech – not by symbolic articulation but through practical adjustment to the world. ...This is knowledge, is it not, in and through active response and adjustment ? The active response and adjustment may be considerably provided for by nature at birth ...*
>
> [Reid61, pp. 15 -16]

The inclusion of experiential knowledge in the modelling process makes the process of EM significantly different from conventional modelling and programming. Propositional knowledge which is represented by symbols is not sufficient to describe real-world situations. As expressed by Reid, *"The symbols express aspects of a given field which is much larger than what is expressed ..."* [Reid61, p. 16]. Knowledge that is based on propositional representation is static and does not spontaneously respond to the changes of one's knowledge gained through experience or interaction with the environment. Therefore the propositional representation system is not well suited to the representation of a dynamic environment, e.g. interactive human activities.

EM is an approach to computer based modelling that is based on new principles for domain analyses and model construction. Computer models constructed using EM principles are not to be viewed as implementing an abstract mathematical model. Their significance is instead that of the model that an expert scientist builds to account for phenomena or that an engineer constructs to prototype or test a design concept. In this respect they are distinguished from the abstract computational operations that happen to be executed on a computer: the manner in which they manifest state to the user is essentially in the interaction that the user can have with them. The state of the computer model reflects experiential knowledge of the modeller or the user. That is, a change of computer state corresponds directly to each interaction done by a user in an explanatory fashion. Visualisation is considered to be an integral part of the model and not just an 'add-on' component. This quality of visibility of the model gives it the

special property that a modeller can experience the model in a similar way that a designer experiences his or her clay model.

Knowledge can be represented in computer-based systems in a similar way to that in which a book represents the author's knowledge. What is special in the EM environment is the following. The way knowledge is represented is based on the 'natural' structure of how people perceive the world, not on abstractions based on the formal structure of logical and mathematical statements (see § 5.3.1). Such a natural structure is rich in meaning and can be comprehended in many ways. In contrast, a formal structure is designed with a fixed meaning in mind.

Points of consideration from 'Radical Empiricism'

The following quote from 'Radical Empiricism' by William James [James12] refers to the possible use of mental models (ideas of reality) in place of their real word referents. The consideration of the possible use of mental models has been extended in EM thinking by way of making them explicit to be able to incorporate actual experiment with them to gain more understanding and perhaps new knowledge. A good understanding can lead to a better practice.

> *By experimenting on our ideas of reality, we may save ourselves the trouble of experimenting on the real experiences which they severally mean. The ideas form related systems, corresponding point for point to the systems which the realities form; and by letting an ideal term call up its associates systematically, we may be led to a terminus which the corresponding real term would have led to in case we had operated on the real world. And this brings us to the general question of substitution ...*
>
> [James12, p. 61]

The EM approach provides an environment where models are working as a platform to construct a substitute for, and an extension of, an active mental model that can be used to share and exchange knowledge. The EM model or Interactive Situation Model as the name implies, is a model that provides an interactive situation in the

problem domain. An ISM can generate computer-based experience that is close to actual experience. The experience-based and open-ended nature of the EM model corresponds to an idea of James that knowing is not a static relation, but that people keep changing what they know continuously [James12, p.75]. For this reason definitive scripts are used to describe the relationships amongst observables (knowledge based on actual experience) and to allow arbitrary (open-ended) interactions.

Views of knowledge in EM perspective

Views of knowledge consistent with an EM approach can be categorised into three main views, namely the first person view, the second person view and the third person view. These three views arise as a result of our recognition of the significance of inarticulate knowledge and the possible loss of this crucial aspect of knowledge when we need to transfer knowledge from private understanding to public understanding.

The first person view refers to personal knowledge by an individual which is particular and provisional. That is, it is how a person thinks and perceives things in a personal way. This kind of knowledge is profound and difficult to comprehensively describe. It resides in the range between tentative knowledge and certain knowledge shown in Figure 7, but is most closely associated with tentative knowledge. The second person view refers to how an EM approach is proposing an alternative approach in computing to allow a user to move from subjective (first-person) knowledge to objective (third-person) knowledge in a computer-based support system. The second person view is concerned with sharing personal views and creating an intersubjective level of knowledge open to criticism. In an EM environment, a representation of activity is made in terms of a coherent correspondence. This refers to the correspondence between the computer-based model and the real world situation under study observed by the modeller or user. This coherent correspondence is achieved by open-ended interaction and the process of observation and experimentation described in § 3.2.1. An EM computer-based model is a representation of experience that represents an actual experience of the real world situation or problem domain. The third person view of knowledge is the kind of knowledge for which current computer-based sys-

tems give most support in representing activity in terms of repeatable observations or experiments, reliable processes or objective kinds of knowledge.

An EM approach focuses on the second person view of knowledge as a significant aspect of using computer-based systems to assist in every day problem solving. This is owing to the fact that in our everyday problem solving we do not use such sophisticated mathematical functions or complicated logical statements as we might need to derive a new proof for a mathematical formula. In everyday problem solving, such as business decision making, we usually do more than observe and investigate the situation of the problem domain in terms of abstract features and their relations. This kind of problem domain needs a different kind of computer-based environment. Such an environment should support observation and investigation, in a way that allows human faculties to be exercised throughout the problem solving processes. This is because only human subjects can have experience and this experience is a significant factor that shapes the observation and investigation.

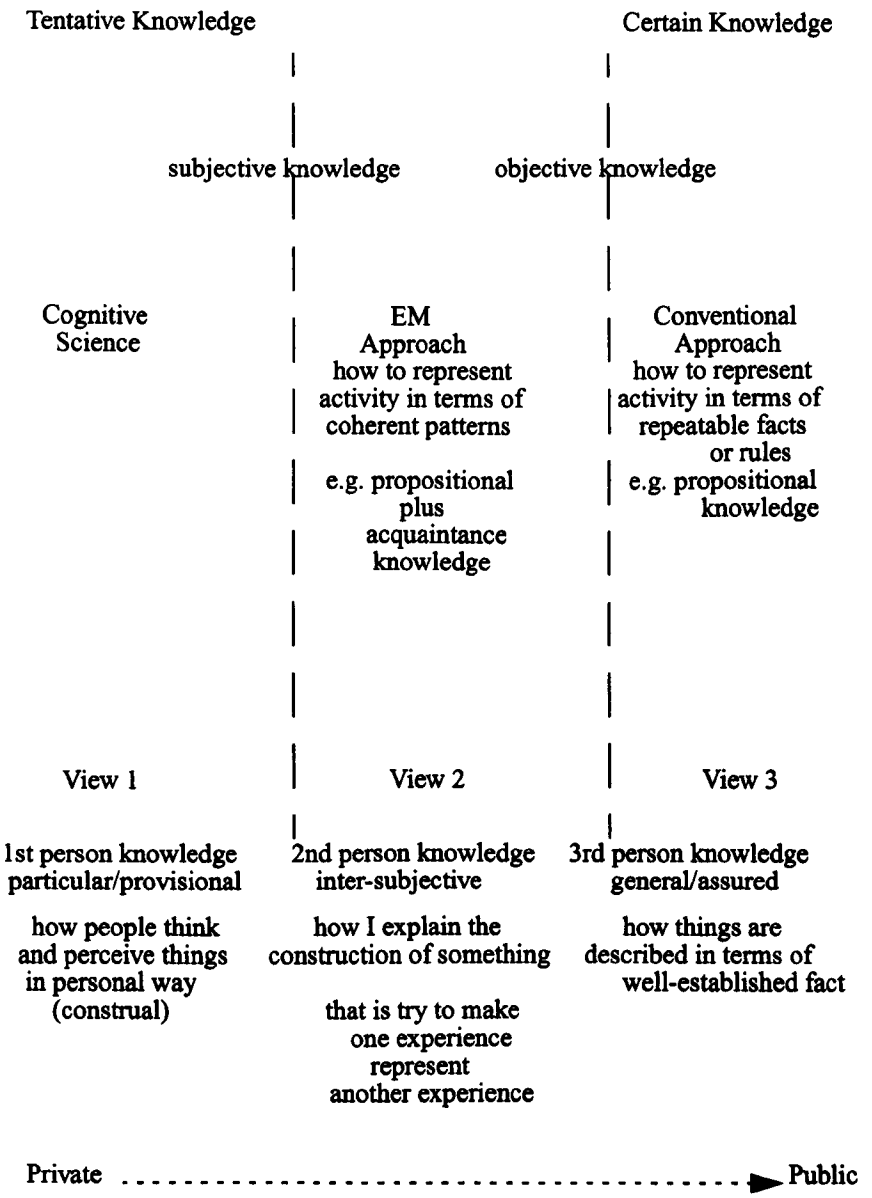


FIGURE 7. Views of knowledge in an EM approach

4.4 Implementation of EM Knowledge Representation

Discussions in chapter 2 illustrate the need for an alternative environment for business computing because of the highly volatile and human-centred nature of the business environment. Then, in chapter 3, the Empirical Modelling approach was proposed as a computing paradigm to support an environment for constructing computer-based business support systems. In this section, how the EM environment supports a computer-based knowledge representation system is discussed in more detail.

4.4.1 Open-ended system versus closed system

EM introduces an alternative approach to system development. Using conventional system development approaches, the way in which a system is constructed is through a *one-way closed interaction* (OCI) while the system development within EM environment is through *two-way open-ended interaction* (TOI).

With OCI, the system under consideration is conceived as bounded by a required functionality. After the development processes of analysis, design and implementation have been completed, major structural change is not allowed. On the other hand, in the TOI approach, the system is considered to be relatively open-ended. The system boundary is flexible and permeable which allows a changeable system structure. In this approach, the assumption is that while the system is being developed the model builder's knowledge is also growing. This will change the model builder's mental model and this will be reflected by changes in the developing system. Such change is rather limited with the use of the conventional one-way closed interaction system approach. The discussion of mental models and how EM can facilitate development of similar kinds of mental models follows in section 4.4.2.

Figure 8 illustrates how an EM environment can facilitate system development in an open-ended way. The boundary of the domain environment or, in other words, the referent system, is represented by a dotted line rectangle. The referent system is the system being studied. The double-headed arrow refers to a two-way interaction. This

means that, while the development process is taking place, the interaction between the model builder and the referent system is still going on until the system construction reaches the point where it satisfies the intended purpose. The two-way-interaction also means the modeller's interactions embrace incremental model building and incremental model interpretation. That is, throughout the construction process, via the interactions, some properties may emerge that modellers find useful for a system and some may be discarded, hence the design of the system is shaped through a better understanding. In addition, such interaction is always available and keeps open the opportunity for further modification of the system if, after some use, the referent system is found to have been changed. The three layers of free shape drawing represents the cumulative structure of the system being built. This is the implication of the growth of understanding and knowledge. Through such a free interaction with the system, the model builder develops more and more understanding and can construct a purposeful system.

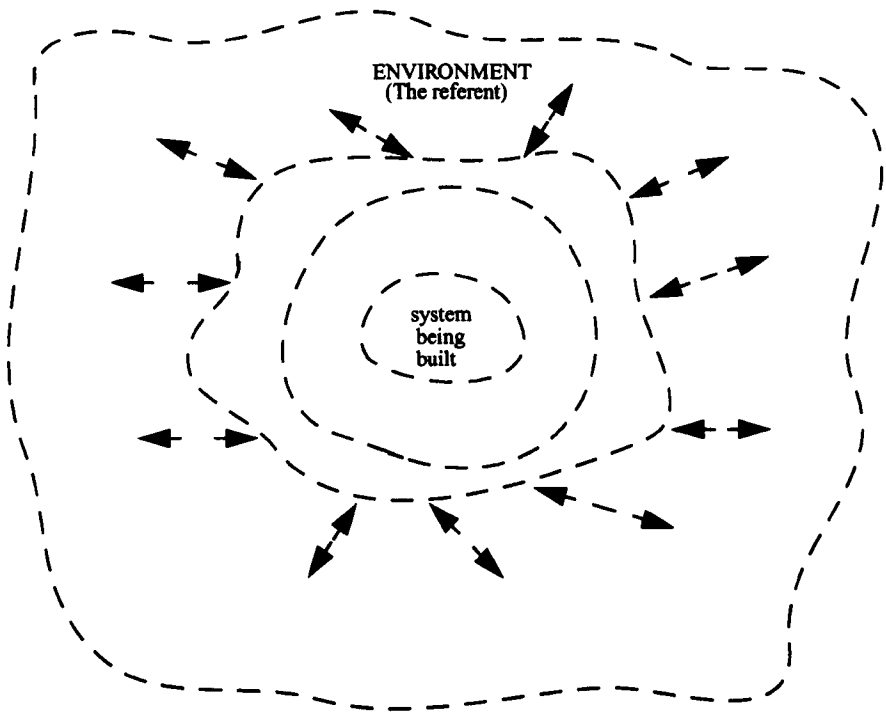


FIGURE 8. EM system development environment: Two-way Open-ended Interaction (TOI)

Figure 9 illustrates the environment of conventional system development. The boundary of the environment is considered to be the same as that discussed above. The differences are the interaction between the system and the model builder, and, the boundary of the system being built. The single-headed arrow represents one-way interaction. This means that during the development process, there comes a point after which there is no further interaction between the model builder and the referent. A single layer solid free shape drawing represents a rigid structure of the system which can be completely constructed only once.

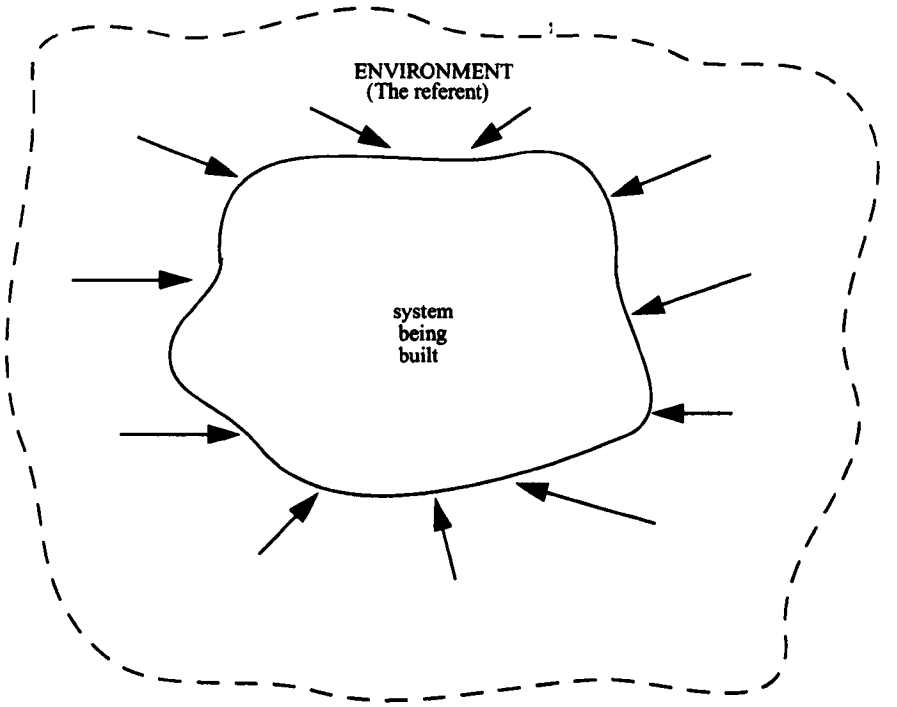


FIGURE 9. Conventional system development environment: One-way Closed Interaction (OCI)

In any computer-based model development, one has to start with a main idea of what is expected from the computer-based model. This is especially true of traditional model development in which every detail of the pre-defined system abstraction has to be completely decided before constructing the model. It even applies to some degree to systems developed using spreadsheets that will serve well only for those transaction decision problems that can be completely specified, i.e. the range of inputs, the

expected range of outputs and the relations between each parameter within the problem domain under consideration. Within EM, even though the major system abstraction also needs to be decided before constructing the model, the need is like the artist's need to sketch or outline their work and then add in their creative ideas in every single detail to correspond to what they expect their work to look like. Models developed by EM are developed more in the way that the artist works (cf Appendix A). This practice of cumulative and iterative development is the natural way for most human intellectual and skill development. The traditional system of development imposes on the modeller or designer the demand to define their requirements in detail in advance. This makes conventional system development difficult and only experts can do such jobs. In addition, human beings experience, learn and re-learn things. It can happen that after a period of exposure to the early design of a system for a certain task, it might be found that the system does not suit the need any more since the requirements have changed. When this happens, it is difficult for conventional developed systems to reflect such changes. EM opens a new way for adaptive computer-based model development when it is relatively easy to reflect such changes of requirements.

4.4.2 Modelling instead of programming

A conventionally developed system enables changes to be made to the values of the data but not so easily to the data types or the data structures. In addition, conventional approaches also make changes in the formulation of solutions difficult although changes of data type, data structure or the solution formulation are common in the business environment. The restriction of changes in conventional approaches are due to the fundamental principles imposed in programming. By way of modelling, the spreadsheet environment is a good example which demonstrates how the modelling principles of *open interaction* and *dependency maintenance* which lead to *automatic updating* enable an environment for constructing a personalised system or end-user computing. Nevertheless, as discussed in § 2.4.2, the spreadsheet environment is much

influenced by the computing paradigm that focuses on utilising computer power rather than promoting human capability.

In terms of modelling activities, spreadsheet systems have two modes of usage that can serve different purposes. Firstly, users may construct a common format of model, such as a finance manager may create a model of financial budget and then implement *what if ?* analysis with variables. Secondly, users may construct an individual model, for example, a teacher may create an animation of the traffic light to show its operation for a school child. Such a personalised system or 'end-user computing', is a system that suits business people the best, as the business conduct of an individual, either at an organisational level or at a personal level, is unique and no generalised algorithm can prescribe it. An EM approach has more to do with this second mode of usage of spreadsheet systems and being able to generalise their properties.

Mental model. According to Norman [Norman83], *"... mental models provide predictive and explanatory power for understanding the interaction ..."*. Norman points out that, *"In interacting with the environment, with others, and with the artifacts of technology, people form internal, mental models of themselves and of the things with which they are interacting ..."* [Norman83]. Norman argues that mental models have an evolving nature. These ideas of Norman are in line with our belief that people learn new things through interactions, in other words, through experience. Since our knowledge is continually changing, the more adaptive systems are the more useful they are going to be as supportive devices in problem solving or decision making processes

How expertise can be explained in terms of mental processes. Miller, cited in [Klein+95], introduces the term 'chunk' to describe the units stored in the working memory. A 'chunk' is a semantic entity – an organised unit of information. The brain's capacity to handle information is affected by the number of chunks in the receiving stimulus. He also showed that stimuli received are organised into chunks according to previous experience. This leads to the assumption that an expert learns to organise more information into a chunk than do novices. Miller's research shows that a person does mental activities – information processing – when receiving stimuli.

This was a change from the behavioural tradition that looked at the brain as a black box.

An EM environment enables the translation of a mental model into an explicit computer-based model that enriches experience through interaction and experimentation with the model. By way of experiencing things, we learn and obtain knowledge. And we can do this with an interactive model as well as with the real world. In other words, EM supports the evolving nature of people's mental models. These mental models are the partial representations of human knowledge. This is part of our argument that EM environments support a new kind of computer-based knowledge representation.

Power of pictures: EM and its use of a visualisation in its basic model structure. Humans engage in imaginative, creative, or analytical thinking, and this is often accompanied by pictorial images. Such pictorial images may be 'in the mind' or may be physical. Pictorial images as a means of representation make sense to many people. Historically, pictorial images were used as the first and the most primitive communication medium between people. Evidence of this, for example, is the naive drawings of people, animals and plants in the stone-age, found in different places around the world. Another example to support this argument is a teaching technique used in kindergarten schools. In arithmetic classes, teachers will use pictures, for example, two units of apples for each set of apples together with a plus sign, to show the operation called 'add'. The abstract operation 2 + 2 = 4 is illustrated by two apples plus two apples equals four apples. This example shows that an abstraction such as an arithmetic operation is not so immediately familiar to people.

EM introduces a use of modelling as a means to construct a system, either to understand the situation or to have built upon such a model a purposeful system. This use of modelling involves visualisation, or a pictorial interface to represent the subject of interest. This use of visualisation to represent the subject of interest is similar to the way in which humans often have pictures in mind when they are thinking

either creatively or analytically of something. This argument is supported by the following quote of Ian Stewart.

> *Some mathematicians, perhaps 10 per cent, think in formulae. Their intuition deals in formulae. But the rest think in pictures; their intuition is geometrical. Pictures carry so much more information than words. For many years schoolchildren were discouraged from drawing pictures because 'they aren't rigorous'. This was a bad mistake. Pictures are not rigorous, it is true, but they are an essential aid to thought and no one should reject anything that can help him to think better ...*
>
> [Stewart75, p. 5]

Pidd [Pidd96] argues for a similar idea. He refers to the work of a bridge engineer who may be working from a *mental image* of the bridge where the appearance, size and load, for example, will be envisaged. Pidd refers to these as implicit mental models. These implicit mental models will be transformed into formal and explicit models which lead to the construction of a physical bridge which meets the requirements very well. The use of different types of modelling is referred to by Pidd as ways of managing complexity.

4.4.3 Integrated environment

An EM approach offers an environment that is integrated in wide-ranging ways including the following: i) integration of human sense-making and computer data manipulation activities; ii) integration of analysis, design and implementation phases of system development; iii) integration of HCI and system development; iv) integration of hard and soft components of the referent and v) integration of kinds of knowledge.

Integration of humans and computers

The following quote by I. Stewart [Stewart75, p. 4] supports the argument that the formalised methods of typical computer science are not necessarily the only suitable way

to develop computer-based systems. This is because not everything can be formalised. Stewart points out that, *"No one has ever formalized significance. To recognize what is significant you need a certain amount of experience, plus that elusive quality: intuition ..."*. This is what EM is concerned with. In particular, EM is concerned with those systems where human ability cannot be entirely replaced by the capability of any computer-based system, but human ability together with the capability of computer-based systems can support better performance.

The other obvious examples of the deficiency of generic programs are the spelling and grammar check features in word processing software. Those features provide only a general check on whether the words are correctly written or not. It cannot spot whether the words are wrongly used. For example, such features cannot distinguish confusion of the word 'effort' with the word 'afford', or use of 'effect' instead of 'affect' in sentences. This example shows the significance of possible mistakes that can be produced if we were to rely too much on a computer-based system without incorporating human sense-making. Such a generic program cannot afford to incorporate every possible aspect of context. There are too many possible elements to be included in the program in relation to both time and cost. Human beings sort out the semantics with much less time and cost compared to computer systems. The human and computer integration particularly suits a context sensitive operation.

Figure 10 below describes an EM environment that enables the integration of human capability with computer functionality. This refers to the *learning through experience* capability in human beings and the *speed and accuracy of systematic analytical* ability of computer systems. Figure 10 illustrates that throughout the modelling process, human agents gain new knowledge while freely interacting and experimenting with artefacts. Artefacts refer to models having geometric interfaces that represent either situations or systems with which the modellers are dealing. This new knowledge is referred to as insight. With insight human agents can penetrate situations or systems much better. Such an integration leads to models that are products of this modelling process and can be further expanded or automated to be used as actual systems. This is due to the nature of this modelling approach that enables the re-applica-

tion of understanding gained through analysis of the system components. The diagram (Figure 10) should not be regarded as two sequential phases (separated by dotted line) but rather as processes continuing concurrently. The part above the line goes on 'within' the part below the line.

Integration of kinds of knowledge

Articulate and inarticulate knowledge both exist in an EM environment. That is, there can be propositional knowledge (such as is represented by conventional computing methods) but also tacit and experiential knowledge are two kinds of inarticulate knowledge that can co-exist with the propositional in an EM environment. This is because the human subject is considered to be a crucial and integral part of the system. Without major intervention from human subject, an EM model can be used as a tool in the same way as a conventional system is used. Therefore, the use of an EM approach is most significant in the situation where the intertwining of different kinds of knowledge is crucial as in the new use of decision support systems that is proposed in this thesis.

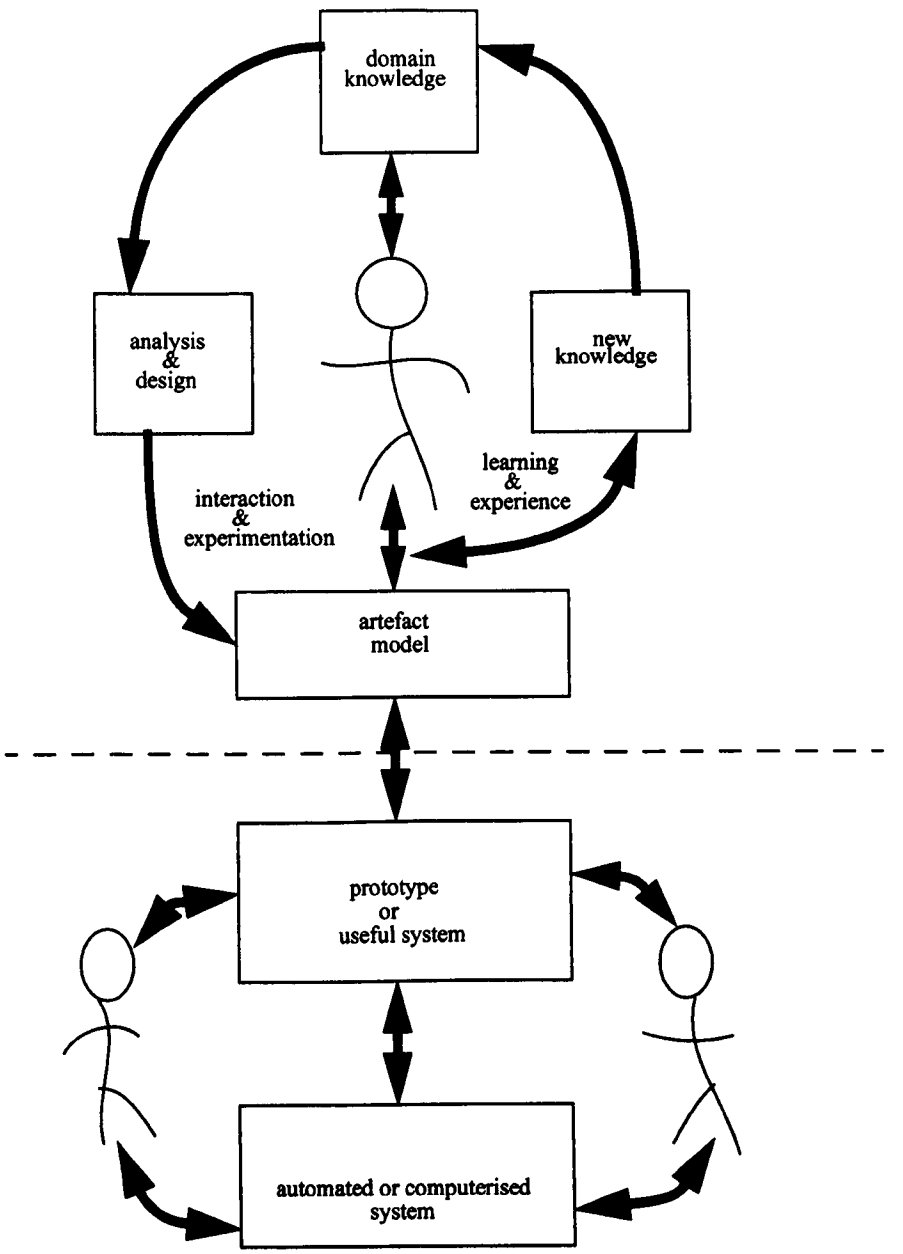


FIGURE 10. An integrated human and computer environment

Integration of analysis, design and implementation phases of system development

In an EM environment, there is no clear separation between each phase of system development. That is a development process in EM is more like the building of a house by a carpenter who knows all the processes very well from start to finish. Firstly, he or she has to do some analysis with the house owners of what they want. Secondly, he or she will hand over a sketch of the design and if they are all agree, then the actual building or implementation is started. During the process the house owners may request to alter some plans and a carpenter may agree to do so easily. This is unlike a production line in a car factory where each car assembler would know only how to fit their particular parts to the body of the car. Besides, a car designer does not typically actually know how to build a car. However, the benefit of each method is at opposite ends. At one end there is end-user appreciation and at the other end there is a commercial product.

Integration of HCI and system development

In conventional system development, most often the interface between the user and the computer system is developed separately from the rest of the system. This often means that users have to adjust themselves to what is available, which may be unpleasant or not what users expect. In an EM environment, the interface between users and the computer system is built as part of the process of systems development. This is because the nature of open-ended interaction allowed in an EM environment supports users experimenting to find out suitable protocols for interaction to operate a system in response to users' requirements. The process of experimenting to seek for such protocols in an EM environment often suggests a suitable form of interface.

Integration of hard and soft components

There are many uses of the terms 'hard' and 'soft' in reference to components of a computer system. Firstly, consider the hard and soft data as referred to by Robson [Robson97]. The hard data are all the data that are needed for a particular process, for

example, data collected from a market survey or a sales transaction report which is stored in a digital form for computer processing. This kind of data is data that can be processed by computers. The soft data is data known by an individual human subject on how, and in what way, to use such hard data. Secondly, there is the combination of hard and soft approaches as referred to by Pidd. That is, the soft approach is an approach in which the construction of a business model is based on the qualitative judgement of a human subject. The hard approach is the use of quantitative modelling constructed by formal methods and theories. The use of both hard and soft approaches are relevant and important in an EM environment where they can be integrated and represented together. The following Figure 11 illustrates the idea.

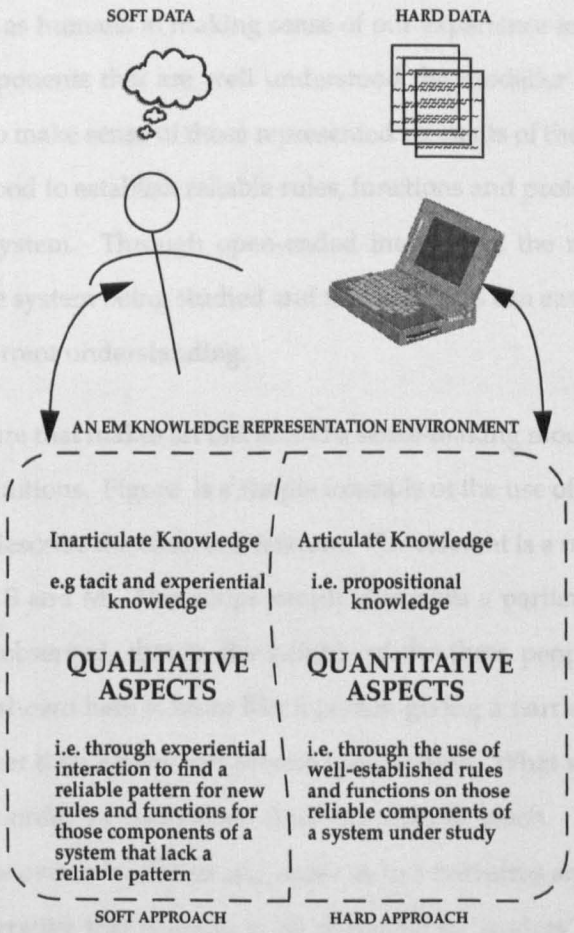


FIGURE 11. Integration of hard and soft components

4.4.4 A sense-making model: the Interactive Situation Model (ISM)

The models developed by an EM approach are basically constructed from common sense knowledge of the domain, as understood by the modeller or user. Such common sense knowledge is then represented by the use of definitive scripts to describe observables, dependency and agency. *Observables* refer to the entities and the attributes of entities that have been observed by the modeller. *Dependencies* refer to the relationships among those entities. *Agency* refers to the agents involved in the system and how each of them can influence the changes of state of entities. The modeller is an individual with knowledge of the domain. Such EM models are sense-making because the basic concepts used in this making (observables, dependency and agency) are those we use as humans in making sense of our experience in the world. Starting from initial components that are well understood the modeller or user experiments with the model to make sense of those represented elements of the referent that are not yet well understood to establish reliable rules, functions and protocols that may apply in the referent system. Through open-ended interaction, the modellers gain more knowledge of the system being studied and the modellers can easily adjust the system to reflect their current understanding.

Another feature that makes an EM model a sense-making model is the explanatory power of the definitions. Figure is a simple example of the use of definitions in an EM environment to describe the observed referent. The referent is a room containing three people named J, S and M. This script simply describes a particular state of the phenomenon being observed, that is, the heights of the three people. The style of the definitive script shown here is more like a person giving a narrative of what is going on in a room rather than a fixed and precise specification. What we mean by narrative style is that the order of definitions does not matter much. You can begin your description of the events in almost any order as in a narrative style of writing. However, this is a narrative that is going to be presented to 'readers' which are computer systems, it needs a great deal of vocabulary that will be 'understood' by computer machines.

```
l = [1,2,3];


// we observe three people in a particular order with names and heights

n = ["J", "S", "M"];

J is n[1]; S is n[2]; M is n[3];

htJ = 165; htS = 160; htM is 71*ins;

ins is 2.54 * cms; cms = 1;

currperson is J;

orderht is ord(htJ, htS, htM);


// how to put group of two, or three comparable things into ascending
order

func min { para x,y; auto result; result = (x>y)?y:x; return result; }

func max { para x,y; auto result; result = (x<y)?y:x; return result; }

func ord { para x,y,z; auto result; result = [min(min(x,y),z),
min(min(max(x,y), max(y,z)), max(x,z)), max(max(x,y),z)]; return result;
}

people_in_the_room is [J,S,M]; hts_people_in_the_room is [htJ, htS,
htM];


// how to get the index of someone of a particular height

func isht { para ht; auto i, result; for (i=1; i<=3; i++)
if(hts_people_in_the_room[i]==ht) result = i; return result; }


// how to apply a function to all elements of a list

func maplist { para l, f; auto result; result = []; while (l!=[]) {
result = result   [f(l[1])]; shift l; } return result; }


// how to put the people in the room into ascending height order

ixordht is maplist(orderht, isht);

func getinorder { para l, ixl; return [l[ixl[1]], l[ixl[2]], l[ixl[3]]];
}

usinorderofht is getinorder(people_in_the_room, ixordht);
```

FIGURE 12. An example of a script of definitions in an EM model

4.5 Conclusion

The foregoing sections have put the emphasis on experiential aspects of knowledge. An EM environment, in which human cognitive processes are within the scope of the data manipulation activity, can hardly be denied the status of being a form of knowledge representation. EM models may be thought of as extensions of people's mental models. The concept of computer systems being perceived as cognitive artefacts supports the use of the computer-based systems as instruments to assist users to gain a better understanding of the situation or the problem domain. Polanyi has said, *"... human beings have enormously increased the range of comprehension by equipping our tacit powers with a cultural machinery of language and writing ..."* [Polanyi83, p. 91]. Cognitive artefacts can be considered to be yet a further major expansion of our powers of understanding and communications.

Focusing on 'tacit knowing' and 'experiential knowledge' which are subjective matters, should not bar an EM approach from being viewed as 'scientific'. Referring to Polanyi, the contribution that a scientific approach can offer to the community consists of three factors: *"... its exactitude, its systematic importance and the intrinsic interest of its subject matter ..."*. He argues that, *"... the proportion in which these factors enter into scientific value varies greatly over the several domains of science: deficiency in one may be balanced by excellence in another ..."* [Polanyi83, p. 66]. Though EM models do not aim initially at objective exactitude they have the potential for faithful rendering of individual experience. EM models are constructed by being based on systematic procedures and their core contribution is their detailed, observational correspondence to real-world phenomena. In an EM analysis, an attempt is being made to offer an insight into the nature of reality; to understand it well and to apply what has been understood to serve the needs of achieving a better solution to a problem.

The reluctance of software developers, or computer system designers, to give proper recognition to the cognitive aspects of computer artefacts has had a long history. In the late 1960s, Cross [Cross67] pointed out that there is a basic mismatch between the computer and the manager. He refers to the problem that, *"... computers*

tend to portray their results in a digital manner, and people tend to perceive in an analog manner ..." [Cross67, p. 14]. This basic mismatch seems to have been an obstacle throughout the whole period of computer systems development. Certainly, there are numbers of significant improvements in terms of graphical and pictorial representations to help alleviate such a mismatch. However, the search for computer-based systems that correspond to human perception and cognition is still biassed towards what a computer can do, not on how we can benefit from the incorporation of computing powers and human abilities. Cross states that such a basic mismatch is the result of long established management processes prior to the computer era. This thesis endorses Cross's concept that there is a basic mismatch between the computer and the manager. But the study does not indicate that this is the result of management processes. Instead our study is critical of the conventional use of human-computer interfaces, of shortcomings in system development and of limited kinds of knowledge representation all of which inhibit use by business people. Such difficulties contribute to the mismatch to which Cross refers.

Chapter 5

A Natural Environment for Business Solutions

5.1 Introduction

In the previous chapter the significance of Empirical Modelling as an approach to the construction of computer-based systems emphasising the role of knowledge representation was addressed. This chapter begins with a view of system development in terms of standard software engineering approaches, where detailed analysis and precise specification are emphasised, and alternative approaches, where human factors and interpretation are of equal concern. The issues which arise in business-oriented system development are then considered, the chief of them being that unpredictable human factors are so central to business domains that applications cannot be built for them using the same tools of analysis and design that have worked for scientific applications. Many alternative approaches to methods of analysis and design have been suggested such as Checkland's Soft System Methodology (SSM) [Checkland81], Pidd's interpretivist approach [Pidd96] and Crowe's constructivist approach [Crowe+96]. The main motivation for EM from a business perspective is that it is an alternative to the engineering approach but the principles of EM are broad enough to embrace both of them.

The central claim of the chapter (§ 5.3.1) is that EM offers a development approach that is 'natural' in the sense that it is grounded in everyday life experience rather in the experience of scientific or specialised practice. The methods of analysis and construction in an EM approach are those familiar from our ordinary experience of making sense of our surroundings and building on what we already know and understand to accommodate new knowledge and make more complex systems. The

fact that the principles of EM are familiar from everyday life and social interaction is what makes this approach well-suited to business applications. How the principles can be applied to building a small business model is illustrated through a case study of the Restaurant Management Model (RMM). This is introduced and used in 5.4 and 5.5.

5.2 Software System Development for Business Solutions

5.2.1 Difficulties of the engineering paradigm

There are three distinct concepts connected with the notion of 'software system'. There is software as such, the need for some process of software development in order to build a software system, and there is software engineering as a particular response to the problems of software development. That response was made in the context of the so-called software 'crisis' and brought the engineering concepts of designing artefacts to perform a given function efficiently, and in a way fit for purpose, into prominence in the production of software. One consequence of this was to make methodologies for development important for the management of software projects. A whole variety of methodologies have been advocated, taught to students, and adopted by industry; they are the subject of much controversy. Particular methodologies have been based around certain models of software development which are still in current use, for example, the waterfall model [Figure 13] with phases of requirements, specification, design, implementation, testing and maintenance. These phases need to be iterated with feedback from ones later in the process. Such iteration is also reflected in the 'spiral model' of development in which each phase is visited possibly several times. There are innumerable versions of the phases and the methodologies. And many technical software tools and environments have been built assisting in the production of the vast amount of software that is being used to maintain current information, communication, and transaction services. Such conventional system development, using methodologies or not, offers reasonably reliable systems to perform well understood,

routine tasks. However, as the structure of the systems demand detailed analysis and precise specification, most often they require a long period of development and are not flexible enough to be adapted easily to changing needs.

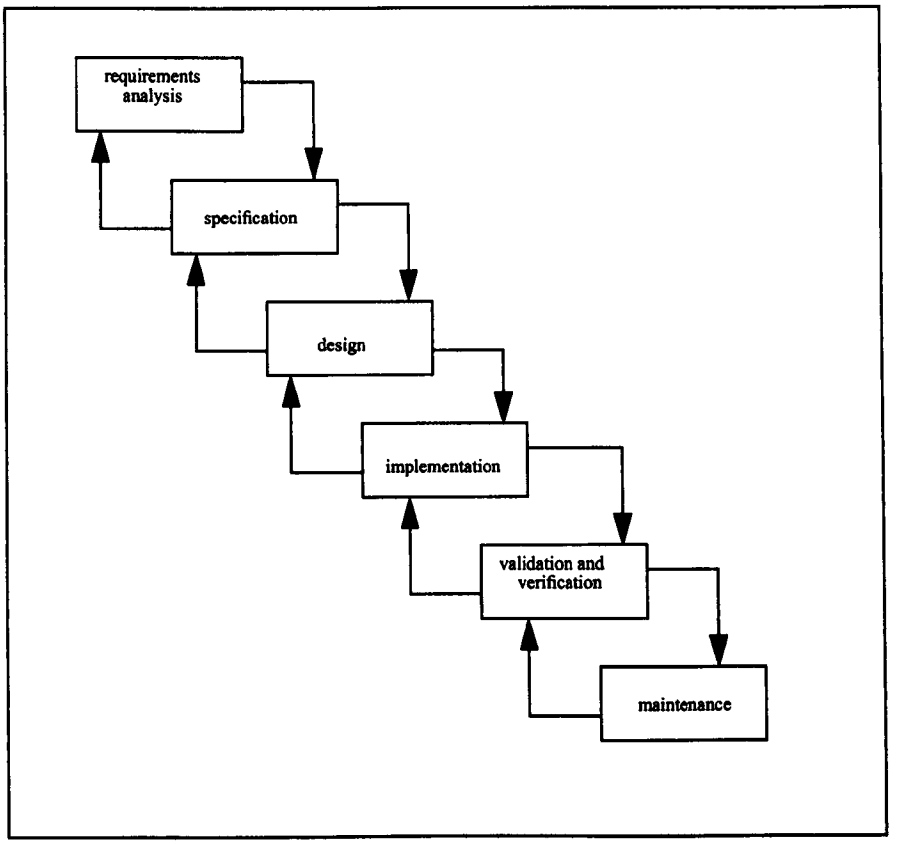


FIGURE 13. The waterfall model of system development
Source: Wilson and Clark [Wilson+01]

To overcome these drawbacks a more recent approach called Rapid Application Development (RAD) [Charatan+01] has been created. RAD is a consequence of object-oriented methods of software development, it proposes integration of analysis, design, and implementation phases of the software life-cycle as shown in Figure 14. In the analysis phase which creates a problem-oriented model, each object is a counterpart of a real world entity being modelled which is capable of providing services to, and requiring services of, other objects. In the design phase, a problem-oriented model is transformed into a computer-based model via a mapping process. Finally, the

objects are implemented with an object-oriented programming language. This integration process is believed to lead naturally to a modular system, and support the criteria of data abstraction, encapsulation and information hiding. [Wilson+01] However, these very criteria are part of the problem when it comes to faithful modelling of real-world applications in which people are involved (see below).

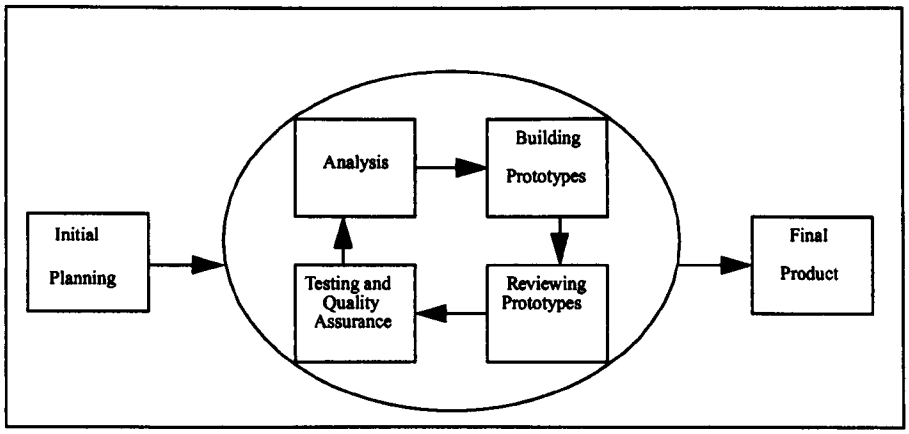


FIGURE 14. A RAD system development
Source: Charatan and Kans [Charatan+01]

In the famous paper, 'No Silver Bullet' [Brooks87] , Brooks argues that at least some of the problems of software development are due to the profound difficulties ('essential' difficulties) of software itself – properties such as its unprecedented degree of complexity and its invisibility – so that the technical tools like high-level languages and programming environments can really only be tinkering with the problems. Even the more major engineering methodologies, or architectures like object orientation, expert systems and connectionism can be of limited benefit. He emphasises the value of adapting what has already been built referring to the organic metaphor of 'grown' software and the strategic importance of design and designers. In 1995 when he republished this paper he found little reason to change his position significantly in the light of developments over the previous 10 years.

Brooks was referring to software development in general. The problems become even more severe when we consider business applications. We have already referred

in § 2.3.3 to some of the problems of software from the perspective of business applications - the artificial nature of the pre-conceived and abstracted entities that programmers typically deal with, the lack of real open-ended interaction, the difficulties in adaptation to different contexts etc. Some of these problems are due in large measure to the influence of the traditional tools and mathematical methods of the engineering paradigm - in using mathematics for analysis and seeking precise specifications of requirements, in using strict data types and seeking for decomposition and modularity in implementation. The particular problems of business where the functions (or requirements) of software are likely to change in ways that are impossible to have foreseen - either because of the environment changing or because the management changes, or both - are emphasised by Avison. He claims that seventy-five percent of software development time is taken up with maintenance [Avison92] and attributes some of the difficulties to the major problem of building models of an organisation as a whole. Even sub-systems are difficult because of the inherent complexity and unpredictability of social situations.

Finally there is the dominance in the engineering paradigm of quantitative and factual information. This is clearly vital for business and decision-making. However, it is obvious that business decision making or problem solving at tactical and strategic levels are not based only on such kinds of information. The qualitative aspect is of equal or greater significance. Such qualitative analysis is often based on observations and interpretations of both formal quantitative information, such as sales statistics, cash flow analysis and sales forecast, as well as informal qualitative information, such as market atmosphere, supplier position and customer satisfaction. The qualitative analysis activity of business managers, consisting mainly of observing and interpreting processes and their relationships, is largely done by so-called mental models of managers without external aids. This often makes the results questionable as such an analysis is regarded as too subjective. The significance of the qualitative aspect in the business environment, and how to incorporate this component while using computer-based support, are major difficulties of business-oriented system development.

5.2.2 Alternative approaches

Part of the reason for the relatively slow development of business applications over the last two decades may be the acceptance of the mathematical, or engineering, influence on software as if it were inevitable, and part of the nature of software. Many of the challenges to how software might be built have come from the business or information systems communities. For example, there have been Checkland's Soft System Methodology (SSM) [Checkland81], Pidd's interpretivist approach [Pidd96] and Crowe's constructivist approach [Crowe+96]. The spreadsheet itself represents an alternative approach to software. The main concern in such alternative software system developments is to deliver an adaptive system that works in a dynamic environment and that is driven by users. A major emphasis is put on the inclusion of users both in the process of development and use [Crowe+96, pp. 6-7 and Rasmequan+00a]. The following are examples of alternative approaches in system development.

A constructivist approach

Crowe et al [Crowe+96] propose a 'constructivist approach' to system development, especially to Information System Development. The underlying thinking is based on 'Systems Thinking' by Von Bertalanffy which, in its early stages was known as 'General System Theory' (GST). The focus of system thinking referred to by Crowe et al is on the emergent properties of a system. This means the properties of a system that arise because it is more than the sum of its individual components. This happens because of the interaction between its components. As they say, *"Identifying relevant individual components whose interaction will produce a desirable emergent behavior is one skill of system thinking ..."* [Crowe96, p. 8]. They suggest that the philosophical ideas that guide systems thinking in engineering, *"... are changing from a historically strong emphasis on reductionist analysis and control mechanisms, to an awareness of softer issues of interpretation, human contexts, and purposes and toward the synthetic and constructive aspects of inquiry ..."*. They believe that this is an approach for handling complex and dynamic situations. They argue that the conventional approach is suitable only to limited domains and applications. This is because, in the conventional approach, a

system is based on the structure with which components have been described in terms of the functions they perform and the interaction between components is often reduced to mathematical relations of various kinds.

In reference to Brooks' diagnosis of the essential problems of software (a section on software engineering and information systems) they suggest five main reasons why contemporary software engineering practice has failed to address these problems. These are as follows [Crowe+96, pp. 6-7].

1. It makes unreasonable demands for requirements to be specified in advance.
2. It fails to match the typical static and unadaptive nature of formulation and specification with the variety of dynamic changes in the interpretive context of use.
3. It seems to invite potential confrontation and misunderstanding. That is, for example, between clients and system analysts, between system analysts and system designers and between system designers and programmers.
4. It tends to be over-reductive in its approach. That is, it fails to take account of the way the information is built up.
5. It lacks a unifying model of software system development.

It is not clear exactly what are the implications of a constructivist approach for implementation. But many of the concerns expressed in chapter 9 of [Crowe+96] such as support for learning, for the enablement of human activity and for incremental development are fully endorsed in practical terms by an EM approach.

Empirical Modelling for software system development

ISMs are a preliminary environment for any system being developed. They can be used in two ways. First, they can be used as prototypes of systems to be further developed by conventional ways of programming. Second, an ISM can be extended in its own right to serve as a system that requires a substantial involvement of human intervention. The basic process for the construction of a model, or a system, in EM has been described in § 3.2.1. There is much ongoing current research on how EM can be applied to software development. Here a brief outline of a process for software system development according to Empirical Modelling principles is proposed and illustrated by Figure 15. Clearly, the EM process of system development bears little relation either to the waterfall model or RAD. That is, the domain under study may be initially treated by an LSD account. The LSD account (see § 3.4) is a basis for developing definitive scripts of related kinds, namely: EDEN, DoNalD and Scout to form a computer-based model.

The process of modelling through a special kind of open-ended interaction in an EM environment allows the incorporation of qualitative aspects which is a distinguishing feature of EM. That is, the modelling of EM is based on experience of the referent and renders a comparable experience through a model. A qualitative aspect of a domain, e.g. a value judgement, or perception of an aesthetic or human feature, is something that can only be experienced not quantified in abstract values. In EM there is an integration between the faculties of the modeller, such as observation, interpretation, judgement and awareness which are subjective and depend on individual experience and knowledge, and the property of computer-based artefacts that enables learning through experience. The output of the modelling process is an artefact which reflects a good understanding of the application. It may be used as an instrument to fashion a variety of potential systems.

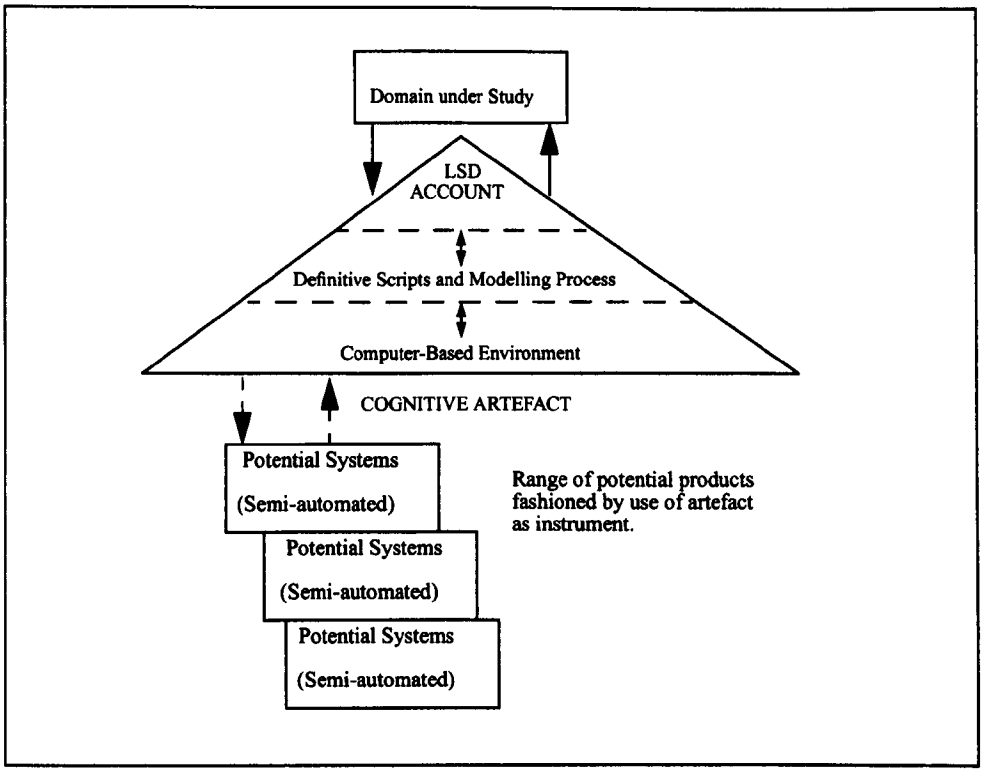


FIGURE 15. An Empirical Modelling system development

The philosophical thinking behind the software system development of an EM approach is distinct from that of the conventional approach. The main focus of EM is on human faculties and shares similar concepts of system development to a constructivist approach, but the underlying principles, development and detailed operation are different. According to Wilson and Clark [Wilson+01, p. 1], in the development of software systems, a programming language is used mainly to help construct reliable software and the development environment is very much influenced by language structures. And it is sometimes difficult to have a clear borderline between the two. This situation is clearly seen in the EM environment. That is, the use of 'definitive notations' is based on underlying principles of observation, dependency and agency which are the structural concepts of the EM development environment.

5.2.3 Complement to conventional approaches

A summary of what conventional approaches have, and what an EM approach can offer as a complement to them is shown in Table 6.

TABLE 6. Two modes of computer use

Aspect of use	Conventional Approach	Empirical Modelling Approach
environment	- suitable to build conventional software product where the environment considered to be fixed and closed - reliability of experience is an important factor of success or failure	- suitable to serve open and volatile environment or where close integration of human and computer processes is required
mode of use	- as a tool, that is, to achieve a certain state and function	- as an instrument, that is, to maintain a relationship between aspects of state
characteristic	- procedural programming possesses a characterisation of a tool by specifying functions explicitly. - declarative programming possesses a characterisation of a tool by specifying functions implicitly	- more emphasis on state and the interpretation of state rather than behaviours and their automation in programming
state	- state described in terms of data types and behaviour in terms of programming abstractions such as procedures of functions	- state described in terms of experience through observables, dependency and agency
tools use	- conventional programming languages	- modelling process uses ISMs' artefacts based on definitive notations

TABLE 6. Two modes of computer use

Aspect of use	Conventional Approach	Empirical Modelling Approach
effect on human cognitive process	- passive use of the computer that is "they are generally designed to exploit the computer's capacity for performing exceedingly complex state-change, and to make the role of the user as clearly defined and simple to enact as possible"	- active use of computer that is a computer system is being used to help expand cognitive process
quality of interaction	- intermittent engagement	1. the **performer** experiences interaction with the instrument as continuous engagement where feedback from the instrument and the environment is involved; 2. the **outcome** of the engagement between performer and technology is more than the accomplishment of a preconceived function; 3. the **performance** will differ according to situation, and be open to influences that are shaped through negotiation and evolve dynamically.

5.3 The EM Environment for Business Solutions

This thesis proposes a hypothesis that an EM environment offers a platform to explicitly represent the process of modelling qualitative aspects of a domain as usually done by a manager's mental processes. As discussed in the previous chapter, an EM environment encourages a special kind of knowledge representation, that is, the combination of propositional, tacit and experiential knowledge, so that the modelling method of EM can support the construction of both quantitative and qualitative aspects of business activities. EM models are used as 'tools for thinking' in Pidd's terms (see § 2.4.2). Modelling using EM is like the way designers develop physical models to represent their mental models of the domain under study. EM models offer a good medium for modellers to interact with artefacts and make qualitative judgements about them in place of the real world domain. Models developed mainly by abstraction are limited in respect of such opportunity.

This is where the definitive notations, used to describe a phenomenon, based on observation, dependency and agency structures, and the integrated environment of an EM approach, are an appropriate means of representing business states. The definitive notations of an EM approach incorporate dependency maintenance and help to provide integrity to a system. The integrated environment of an EM approach, which allows for the integration of humans and computers, provides economy of scale to a system. This is achieved by the human capability of discriminating between choices that normally reduces the great amount of unnecessary processing of alternatives by computer systems. Other technical benefits of EM that are worth highlighting for business solutions are the ability to develop explanatory models in an incremental fashion, and the goal that all model building should be done by the end user. On the management aspect of EM the main benefits are the high potential for reuse of model components and the relatively low maintenance cost because of the correspondence of the modelling process to mental modelling.

The major strength of EM as an alternative system development environment which has its roots in the principle of definitive programming as the spreadsheet does, is that it offers an alternative for business model development, especially in the rarely addressed area of strategic decision support model development. The definitive principles where the value of one thing has been defined in terms of relationships to others, offers a natural way of representing dependencies in the real world. The change in any value within a particular definition (formula) will be reflected in the value defined by that particular definition automatically, without any interference from human agents. This feature allows a more efficient and effective method of model development than those in conventional programming, especially for imprecise and incomplete models. Qualitative aspects of a situation are essential to strategic decision making, and they can be incorporated when the models are developed or used, because of the inclusion of experiential knowledge and the unusual openness of the EM environment. The openness of the EM environment offers the potential to meet the challenge of building models of so-called 'human activity systems'. This term is introduced by Checkland where the social factors within the organization are involved during the development of any system to be used within the organization.

5.3.1 A natural environment

A claim made in this thesis is that an EM approach offers a 'natural' environment for business modelling where one may come to understand better the situation or the problem domain. This is based on the following properties of an EM environment.

1. **Systems are described as seen** – This quality is derived from the three main EM principles on observation, dependency and agency as discussed earlier that are used to describe a situation or a problem domain that needs an analysis. The process of describing a situation or a problem domain in this way, reveals a person's own experience of the situation. This is because the modellers have to identify the fundamental structure of a situation in a way that makes sense to them. In doing this, any EM model is an imitation of a phenomenon as observed. The merit of having this kind of structure on a computer-based system, is twofold. Firstly, it allows a human subject to have a better

comprehension of computer-based models compared to abstraction based models. This is on account of EM definitions that have the form of **'identifier *is* expression'**. With this structure, **'identifier'** refers to an observable element of the system. Secondly, a model is constructed based on their understanding, in other words, their experience and knowledge about it. They can see it, use it, and manipulate it to see for themselves any problem and may try to introduce a remedy or suggest an improvement. Having a chance to test and see results on the computer-based model before implementing any action on the real-world referent brings tremendous benefit.

2. **Systems are seen as described** – In doing an EM modelling process, as discussed above, a person needs to describe a system as he or she sees it. This process would not contribute very much if after all they cannot see exactly what they have described. In an EM environment, we introduce a metaphorical representation for the observables by creating an indivisible link between them and their corresponding visualisation. Therefore any state change made to any observable either by giving a new definition or by redefining the existing one will cause a corresponding change in its representation.

3. **Interactions are made in response to state-as-experienced** – Being able to describe what has been seen and see what has been described supports the process of interaction in response to state as it is experienced. This process allows us to build a reliable model of things we are dealing with. This is because the model comes from an explanatory account that has been experientially verified.

4. **Experiment is based on experience** – In developing software systems in an EM way, as described above, most components are based on the modeller's or user's understanding of the referent. Therefore any experimental implementation looks for reliable patterns of relationships between the computer-based model and the real-world referent and is driven by individual experience. This kind of experiment based on experience supports the quality of an EM environment as an arena for interactive knowledge representation. As discussed in Chapter 4, there are some kinds of knowledge that cannot be articulated and that may appear only in action. This led to our hypothesis that experience-based models are open to include those inarticulate kinds of knowledge that

are different from person to person and probably from firm to firm. Such tacit knowledge may be of the greatest importance.

5. **Reflection on experience gives rise to learning** – This is very much where the integration of kinds of knowledge in an EM environment takes effect. That is, in an EM environment, one can convey one's experience arising from different kinds of knowledge, propositional, tacit and experiential and make it explicit to oneself or others. An EM environment with the use of Interactive Situation Models give the process of reflecting on one's experience more prominence than in a conventional environment.

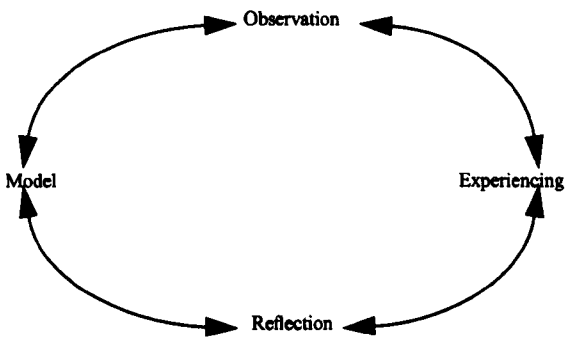


FIGURE 16. An EM cycle of learning

5.3.2 Abstraction-based versus experience-based

Conventional ways of system development use structured analysis and design. Such practices reduce actual experience to abstraction (based on mathematical and logical concepts) that is governed and operated by programming conventions. A. N. Whitehead [Whitehead61] points out that the mathematical world is a world of abstraction that takes things under consideration to be true and consistent in any circumstance without consideration of their context. One of his statements quoted as follows represents this view,

> *... the leading characteristic of mathematics is that it deals with properties and ideas which are applicable to things just because they are things, and apart from any particular feelings, or emotions, or sensations, in any way connected with them. This is what is meant by calling mathematics an abstract science ...*
>
> [Whitehead61, pp. 2 - 3]

Such major use of mathematics and logic in conventional ways of software systems development may suit the conduct of physical science where there is a minimum of human involvement in a system. Examples of this kind of system are the development of a quantum physics model with the use of pure mathematics to find solutions, or the development of organic chemical models using graphics-based on mathematics to represent the composition and structure of molecules of chemical compounds to analyse the possible combinations in order to make an appropriate compound. These kinds of software systems have benefited a great deal from the advance in computing technology in hardware and communication. Increased speed and calculation capacity have had a significant effect on costs as compared with the early stages of the computer era.

The portability of the machine (e.g. lap-top computer) and the high standard of communication networks which can now be easily obtained, for example through a telephone line, enables a flexible exchange of information anywhere in the world. These advanced technological developments make these kinds of systems benefit even more from adopting mathematical abstraction in their system developments: the greater the machine capacity, the better the results, e.g. faster speed and larger memory space enable both the calculation of complicated mathematical equations and the enormous vector structures needed for good quality graphic-based systems. On the other hand, a different outlook is needed in the social sciences where human beings are the centre of most activities, and this demands a different kind of system development. Whitehead clearly points out that mathematics is not suitable for representing human affairs, *"It is natural to think that an abstract science cannot be of much importance*

in the affairs of human life, because it has omitted from its consideration everything of real interest ..." [Whitehead61, p. 3]. Whitehead further points out that mathematical ideas, because they are abstract, "*... supply just what is wanted for a scientific description of the course of events ...*". Human activities cannot be described just by the course of events.

Human activities involve more than a simple explanation of the course of events. Each action or behaviour is often a result of more than a single explicit causal structure as applied by mathematical and logical concepts. Human beings possess the capability of learning and adapting environments to satisfy their needs. Each activity is shaped by its context. Hence, a support system to perform better business activities should be able to take account of each context derived from human observations rather easily. For example, the marketing planning for a new kind of product needs a different marketing campaign in a different market. This is because a different group of people, for example people in a different country, have different likes and dislikes. Therefore consumer behaviour is different from one group of people to another. Of course, there is a theory of consumer behaviour that includes certain constraints, such as income range, taste, life style and the influence of peer groups. However, there are no two markets with identical features. That is to say, abstract explanation which is context-free and assumes things to have generic identity is not well suited to describing human activity, which is a context sensitive phenomenon and very much possesses individual identity.

In constructing ISMs we aim at having computer-based artefacts that represent a real-world referent that is easily recognised when the user interacts with them. That is to say, ISMs or computer-based artefacts make much use of state-as-experienced (see § 3.2.2). This refers to the nature of human experience that varies from person to person and from one circumstance to another. In addition, human beings possess the capability to learn and re-learn from experience. They can relate to things of which they have no direct experience by means of others of a similar kind with which that they have had direct experience and make sense of them. This ability can hardly be achieved by most computer-based systems. The concerns with which particular persons can deal will vary according to their experience. For example, the grid cells used in spread-

sheet systems are exact representations of worksheets or analysis papers used by people in the study of accounting which is a basic subject for any business students. This presumably helped the use of spreadsheet systems to become popular among people in the business field in the 1970s. This example illustrates how the use of familiar representation by a computer-based support system for human activities can promote the extensive use of such a system. Any computer-based system that provides facilities to represent the variation in human experience could serve the purpose of a computer-based support system for human activities more appropriately than the ones which offer only a fixed representation which may not fit well with the experience of many of the users. This kind of system encourages the non-programmer users to feel more comfortable using such digital devices to support their task activities.

5.3.3 ISMs for business applications

A business system is a human activity system. Like any other human activity system, it is hard to describe. This is because of the nature of such a system of which the major component is human beings. The human element is the most sophisticated component of any system because human beings construct individual meanings. Personal meanings distinguish one person from another. We normally never know exactly what to expect from others. For example, when sale managers prepare their proposals for their prospective customers, they can only prepare them based on their best knowledge about such customers. It is often the case that customers make unexpected responses. This means that a number of well prepared proposals fail because they are based on sophisticated quantitative calculations which do not consider human factors to any extent, and these may change the circumstances.

As mentioned earlier, a system which involves human components is hard to describe, therefore, an alternative way to describe such system is necessary. A quote from Minsky [Minsky88, p. 17], *"What can we do when things are hard to describe? We start by sketching out the roughest shapes to serve as scaffolds for the rest; it doesn't matter very much if some of those forms turn out partially wrong. Next, draw details to give these skeletons more life flesh. Last in the final filling-in, discard whichever first ideas no longer fit*

...", reflects both the objective and the outline of the modelling process provided by an EM approach. In principle, one of the major benefits gained by the users of an EM approach is that they have a supportive instrument to describe and derive a purposeful system.

In using ISMs for business applications, a shift is necessary in how the business process is viewed. That is, the business process, specified conventionally as an abstract pattern, needs to be viewed more as a situated activity. **Abstract pattern specifications** adopt abstract computational states, that is formal state-transitional models, and business process protocols which are collections of rules that define human participants' roles. By adopting these two concepts, the interactions and situations referred to in such a business process are presumed to be so well understood that the observations and actions to which they correspond can be specified without considering their particular setting. That is, the intrinsic knowledge or comprehensive experience establishes the validity of the model. In contrast, **situated activity development** takes a view of the business process as an actual experience. That is, the model is derived from the actual experience when engaged in carrying out the business process.

However, in developing ISMs which are computer-based models, unavoidably there is a combination of both abstract pattern specification and situated activity development. Nevertheless, ISMs are more oriented towards situated activity development. In developing models of business processes, ISMs are used to represent the way in which business activities are construed as operating. To seek for more situated models of business processes is essential in order to maintain continuity and stability, two features of the business environment which may be problematic. For example, technologies and working practices change with unexpected and irrevocable events and patterns of activity can be influenced by factors that appear unrelated to them.

The business environment is much more unpredictable than the pure science environment where the very first computer programming principles were developed. The degree of change is more rapid and widespread. This is one of the reasons why the

introduction of situated activity development in Empirical Modelling enables the models to provide an explanation of process behaviour which accords well with a commonsense explanation. We regard the EM approach as a scientific approach for modelling and not a method. This is because, in constructing an ISM, there is no systematic procedure that can predict which experiments it is most appropriate to perform at any point.

Using ISMs as representations of business activities as operating aims: firstly, to model the agents, observables, and dependencies that operate in the business environment, then to explore the way in which these frame the reliable patterns of interactions that underpin effective business processes. Because of this and the fact that the EM approach employs no general method of modelling, there is no way of knowing whether it is possible to give effective answers to key questions in any particular situation. On the whole, in developing a system based on an EM approach benefit would be derived for both developers and users who would gain new insight into the observables, agents and dependencies of such a system. These are significant elements of any system being constructed.

The following are examples of models (computer-based support systems) developed using the EM approach. Each of them illustrates support for human activities which is different from that offered by conventional systems. That is, they emphasise a richer but less optimised manner of integrating devices into the environment: users and real-world situations. In conventional business operation, a narrow sense of integrating devices into the environment is applied. Automatic agents that operate in a constrained environment to which cheaper and more efficient specific functions can be delivered, restrict the involvement of users' awareness of situations at hand, which is a significant factor that distinguishes one business from another [Beynon+00a].

The Timetabling Model. – A screenshot of the timetable ISM is shown in Figure 17. Figure 17 and the model is described in more detail in [Beynon+00b]. It has been developed and tested in actual use for timetabling the oral presentations of about 80 final-year project students at Department of Computer Science, University of Warwick. It does not provide a user with an elegant automated product. But it has been helpful in terms of understanding the situations and problems that arise during the timetabling process and the open-ended experimentation possible allows the discovery of 'neighbouring' solutions when some small modification to an existing timetable is required. This timetable ISM could be readily modified and used for scheduling tasks in business operations, such as shipping lines and distribution chains.

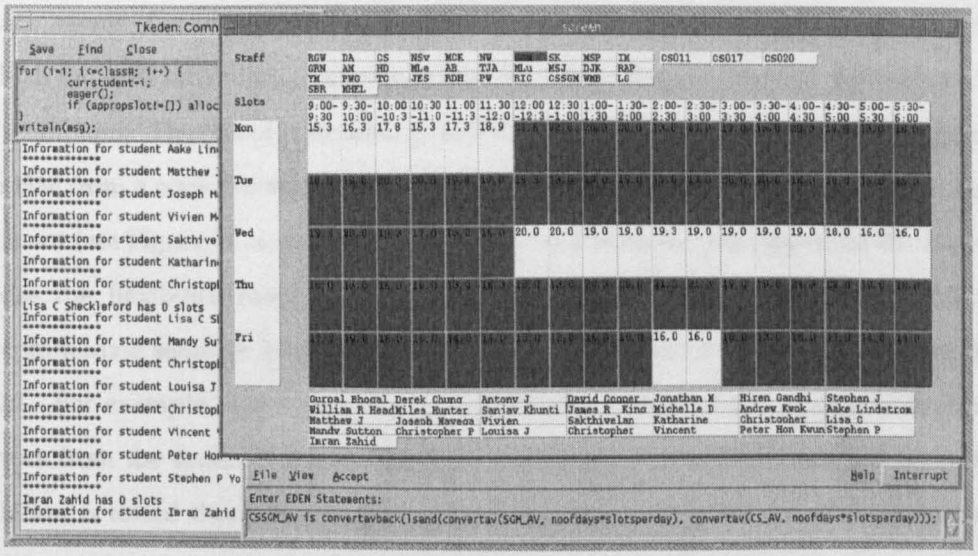


FIGURE 17. The timetabling model

A scheduling task that involves people's commitments is by its nature hard to automate. This is on account of the wide range of variation of interpretation by the people concerned. In addition, the situation may vary because of the influence of the level of interpersonal skill a scheduler may possess. The timetable ISM involves scheduling about the students, each with a supervisor, assessor and moderator, for half hour sessions between 9 am and 5 pm from Monday to Friday. The timetabler has some discretion over the choice of second assessors and moderators, subject to their

areas of competence, and attempts to balance the staff workload. The timetable ISM is best viewed as a form of timetabling instrument to be used to support semi-automated timetabling activity. Its primary role is to assist the manual timetabler in recording and accessing the observables that are of crucial interest in developing the timetable. These include, for instance, the declared and current availability of staff, students, and rooms, the suitability of staff to examine projects, and any problematic consequences of a particular partial or completed timetable, such as instances of double booking or failure to schedule.

The timetable instrument was constructed by combining several simpler prototype instruments, each specified by a definitive script together with several user-defined operators to be used on the right-hand side of definitions, and a few associated automatic agents. One component of the instrument was an ISM that captured the way in which the staff scheduling depended upon the timetabling of students and allocation of examiners. The fact of maintaining such dependencies does not of itself distinguish an ISM from a conventional timetabling tool, but the manner of its construction is significant (see Figure 18 and for more extracts from this model see Appendix B). It provides the flexibility needed to extend the capabilities of the instrument, and to accommodate singular features of the timetabling situation without reinitialising the ISM. For example, whenever there are problems, such as errors arising from data, errors from the faults of the instrument itself, or errors from the user, the semantic relation is tested through experimental interaction until the source of inconsistency has been identified, and the appropriate corrections and refinements can be made on-the-fly.

This process of experimenting by interaction to sort out inconsistent semantic relations illustrates the properties of being experience-based, reliable and human-centred in an EM model. That is, the whole process of developing or using a model is based on domain knowledge that was represented in a model and the sort of interactions we can have with it. Articulate knowledge is transformed into those structures and functions of the model. Inarticulate knowledge is exhibited, for example, in the top-view structure of the model, how it is supposed to function, its interfaces and possible inter-

actions. These three features are gained directly from experience and knowledge of the field. An EM approach encourages users to develop their own models to support themselves in performing their job better as they become aware of the significance of experiential knowledge that is hard to articulate. This is how we believe an EM model can represent the integrated kind of knowledge that is used in everyday life to make sense of situations being modelled (see § 4.3). The right question, or interaction, seems to invite inarticulate knowledge to be expressed in action. This kind of knowledge representation can happen in an EM environment easier than other 'open'[1] systems because of our emphasis on observables rather than the typed variables adopted in conventional paradigms (see § 3.2.2).

```
staff is ["ABC", "OBJ", "PVC",..., "GLC",...]
/*list of staff*/
SAM is [["James Bond", "ABC", "PVC", "GLC"],...]
/*supervisor, assessor, moderator assignment*/

TT is [["James Bond",3]]
/*partial timetable - list of (student name, scheduled slot)*/

ABC_AV is [1,3,4,6,8,...]
/*list of slots that ABC has declared as available*/

avail is [ABC_AV, OBJ_AV, PVC_AV,... GLC_AV,...]
/*declared availability of staff members*/

assign is makestaffassign(TT, SAM);
/*slots so far assigned to staff members*/

current_student is "James Bond";
/*student who is currently the focus of interest*/

S is index(staff, supervisor(current_student,SAM));
A is index(staff,assessor(current_student,SAM));
M is index(staff,moderator(current_student,SAM));
/*S is identifying index for supervisor of current student..*/
avSAM is MEET(avail[S]-assign[S], avail[A]-assign[A],...);
/*possible slots in which to schedule current student*/
```

FIGURE 18. Definitions from the timetabling model

1. 'open' systems here refers to those systems which consider the significance of reflecting changes and incorporating human subjects.

If this model were used for scheduling tasks for shipping line operation or distribution chain operation, the significance of an EM approach that takes account of an evolving real-world situation might alleviate the difficulties for such firms. For example, a firm might need to restructure some of its system components that no longer suited the situation after the events of September 11th, without re-designing its whole system. A company with a distribution chain near the Afganistan border has to go through a complete change of both physical-routes and cost-structure. This is because in conventional systems physical-route structure and cost-structure are usually described independently to maximize the optimum results by, for example, searching for the shortest route to a certain destination. But in EM where we often describe things as we experience them, a definition of the relationship between the route and its cost will be described by an indivisible link and always kept true by dependency maintenance. That is, if the route was changed then the cost will be indivisibly reflected as it is normally observed in everyday life. This type of conduct introduces some reliability when a system is getting more and more complex.

The Warehouse Management Model. – A screenshot of the Warehouse Management Model is shown in Figure 19. It is being developed by Y-C Chen in order to apply similar principles to analysing the business processes that operate in a warehouse application (cf [Sun+99b]). The purpose of this ISM is to enable the blending of abstracted and situated views of the business activity (cf [Turner96]).

This model is an example of many possible models that managers may build to study business processes to either solve a problem, or make a strategic plan for an operation, or a firm as a whole. This model is developed to propose an alternative way of studying the business processes prior to the implementation of actual re-engineering of the process of a firm [Beynon+00a]. This kind of business application of an EM model supports the cooperative culture within the organisation. That is, a business process under study is represented by a computer-based model to show its current status, to be used as a medium to communicate, share and exchange ideas, among

members of staff. A contribution of this model to business activities lies in the quality of the model – it can be used by participants in the process to demonstrate the significance of human-activity and help users contribute to the shaping of a useful system.

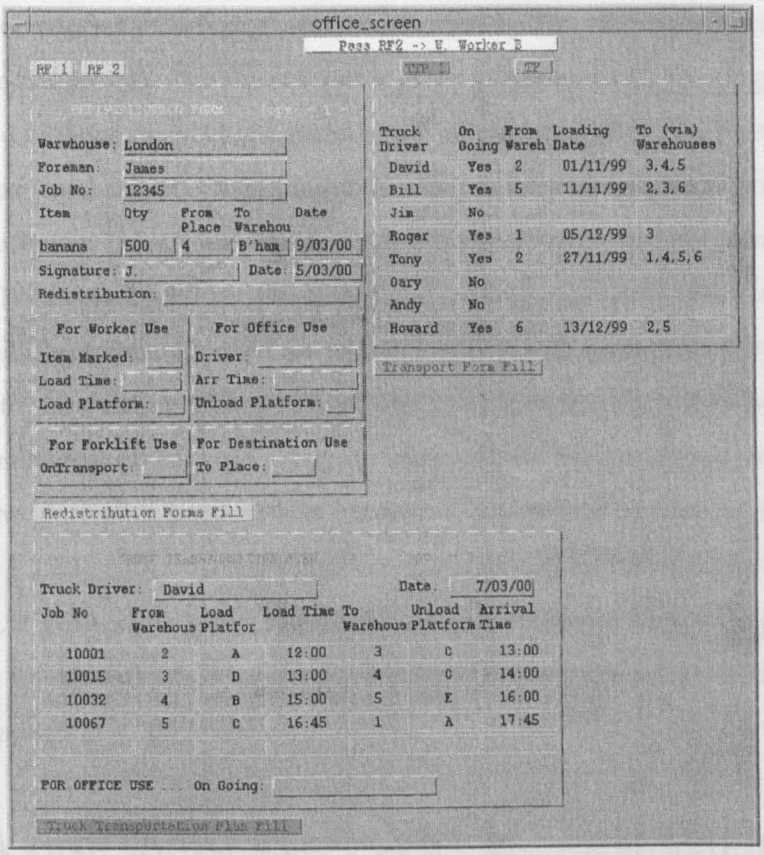


FIGURE 19. The Warehouse Management Model

5.4 Empirical Modelling for DSS

In order to illustrate the ideas and claims put forward earlier in the thesis a case study will be introduced here. It is an idea suggested by this author and some modifications and experimentation have been carried out by this author but the main development work has been done by Chris Roe with modification by Tim Heron (both research students of the EM group). The entire script is given in Appendix C. The original model was devised without any very specific purpose in mind apart from the exercise of visualising the mental model of a restaurant manager as they take or reject bookings for a particular evening. Later it was realised that this could readily become a training tool for the skill of dealing with such bookings and making appropriate table allocations. Chris Roe also subsequently used the model, together with a simple customer spending model to show that it could be linked easily to a data mining tool he had devised to explore relationships between group size, time of meal and amount spent. It is well-suited to the illustrative purpose here because: it is clearly an 'extension' of a mental model (see § 5.4.3), it illustrates the building of a model in an experimental, incremental fashion, it illustrates some of the claims about an integrated knowledge environment from § 4.4.3 and it will be used in § 5.5 to illustrate the principles of how EM could be applied to strategic decision support.

5.4.1 A case study: Restaurant Management Model (RMM)

The RMM as developed by Chris Roe only addresses one function of restaurant operation in a simple way, namely, the acceptance of bookings and allocation of tables. But it demonstrates how the EM environment offers an alternative system development, and it is sufficient to illustrate the main claims from Chapter 4 about the special form of knowledge representation that is possible in such an environment. EM tools offer all the necessary instruments for the model builder to develop artefacts as a representation of a real-world domain and then arbitrarily to explore and exploit the artefacts. It shows immediate feedback (interpreted response to re-definition by the user), indivisible visualization, it allows both definitions and algorithms in the traditional pro-

gramming sense, and the interfaces are developed in parallel with the model building process. These features encourage and shape the human thought process in a similar way to the craftsman producing his or her craft-works. The shaping of artefact(s) by a craftsman represents well how an EM model is developed. In a similar way, there is the same spirit of enjoyment and delight in developing an EM model to represent a domain as there is in developing a craft work.

Unlike the traditional system development where things are regulated, functional and fixed, there is a liberty of thought process and arbitrary experiment which allows the learning process to grow more broadly and insights to be developed. The closest analogy to the process of conventional system development is the process in which an industrial piece of work is produced. In such process there is no room for arbitrary experimentation. The production is governed by rules and restrictions. We have to follow the system process step by step obediently, otherwise there might be a crash in the system with a financial cost. Within the EM environment, all ideas may be tried out with the minimum of expense and there is no serious effect to the system if things go wrong. The lack of artefacts for the system builder to interact with in an open-ended way is a disadvantage of the conventional system, as artefacts for the system builder bring imagination into life.

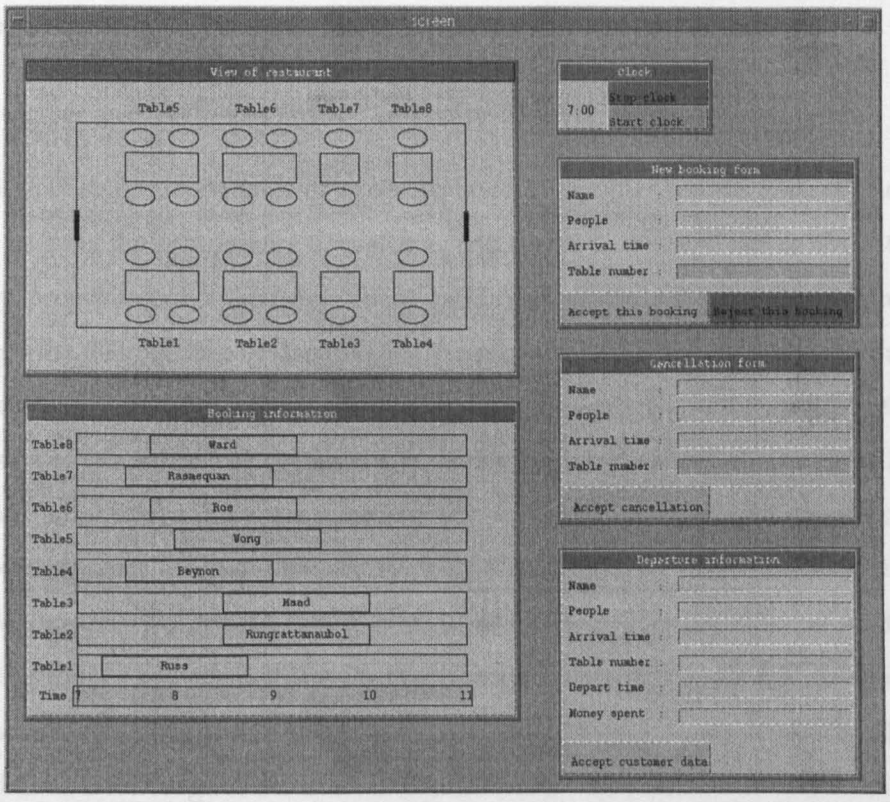


FIGURE 20. An ISM for Restaurant Management

The script in Appendix C consists of two scripts: NEWrest.e and NEWrestwindows.s. On p.C-5 of the former script there is the statement to 'include' the latter script that simply manages the various windows required for the display of the layout and occupancy of the tables, the timetable display, and the windows for the clock, new bookings, cancellations and for collecting final customer data (see Figure 8). The script NEWrest.e begins with definitions in *tkeden* to describe the layout of the restaurant and most of these contain observables with self-explanatory names. They also contain data (in `b1name`, `b1table` etc) for the bookings already made in advance of the evening to be displayed. As with most EM models the user is able to play the roles of both user and modeller and so can interact with the model both through the given interface (e.g. the 'Table number' field in the 'New Booking form' window) and the '*tkeden* Input' window to make direct re-definitions. It would be easy to display a lay-

out of a restaurant plan to scale and for a manager contemplating purchasing new furniture to experiment with the sizes for tables for four persons. For example,

```
R_tablewidth4 = 12; R_tableheight4 = 8;
```

might enable an extra table or two to be fitted in, or other layouts to be considered while keeping the convenience of customer and waiter mobility in mind. While this is clearly propositional knowledge about predicates of the tables it also expresses, through the maintenance of dependency, the modeller's further propositional knowledge about certain relationships holding between the seats, the labels and the relative positions of the tables (see definitions of `pseat2`, `pseat3` and `table1_label` and `table5_topleft`). These are automatically updated and displayed appropriately immediately after accepting the above definitions. There are three-dimensional versions of line drawing systems, and models in which definitions are linked to rendering tools to produce realistic displays. In this way it is reasonable to say the manager can gain some idea of what the resulting table layout would actually look like - this is experiential knowledge of the effect of the new table dimensions. It is easy to imagine all kinds of 'what-if' experiments that could be conducted on a restaurant layout along these lines. The sense in which such a script and display is a generalisation of a spreadsheet should also be clear.

Below the window displaying the restaurant table layout there is a visualisation of the current state of the evening's bookings and their table allocations. One obvious 'function' of the model in its present form is to help a manager in the job of accepting or rejecting bookings according to the tables available. The 'Booking information' window helps in this, particularly for people entering, or telephoning, for a table that evening but without a prior booking. In an early form of the model there were random arrivals or phone calls during the evening to which the user needed to respond. In the version presented here the user can make up a pre-conceived list of customer events (see p. C-13) for the purposes of experimentation with policies, or for training. The random version could also be useful for testing the effectiveness of a given policy. In the present form, when the clock is started there is a compressed animation of the

entire evening's arrivals and departures in the upper and lower main windows. Places in the restaurant are coloured green when occupied as are the booking slots in the lower window. A blue line traverses the lower window to indicate the passage of time, and, in the absence of any events the user needs to attend to, the whole evening is traversed in about 20 seconds.

The goal of the manager might be to maximise occupancy (or income) over an evening without loss of customer satisfaction. Because it is possible to put tables together this goal becomes a kind of dynamic timetabling problem as the resources (tables) are changing during the evening as well as the demand. In order to model putting tables together, for example, putting Tables 7 and 8 into a new Table 7 for four people, we could use the following definitions:

```
%donald

table7_topleft = R_bottomleft+{57,R_height!-10-R_tableheight2!}

/* Note that by dependency then the complete table and the
associated places defined by observables pseat21 and pseat22 all
move appropriately. */

/* Also the seating configuration defined by seating_arrangement
= [ [1,2,3,4], [5,6,7,8], [9,10], [11,12], [13,14,15,16],
[17,18,19,20], [21,22], [23,24] ]; must be re-defined to */

%eden

seating_arrangement = [ [1,2,3,4], [5,6,7,8], [9,10], [11,12],
[13,14,15,16], [17,18,19,20], [21,22,23,24] ];

/* so that the model recognises this new combined table as one
suitable for four. Then also to correct the labels for this
arrangement. */

%donald

table8_label = label ("",table8_topleft+{R_tablewidth2!div2,
R_tableheight2!+15})

table7_label = label("Table7",table7_topleft+{R_tablewidth4!
div 2,R_tableheight2!+15})
```

These changes now result in the layout in Figure 9.

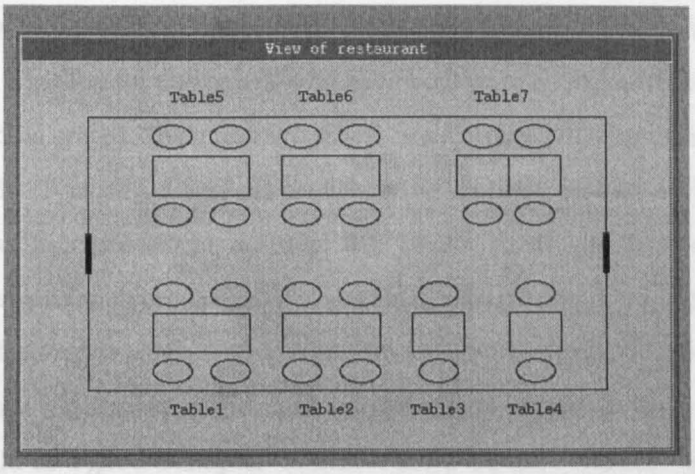


FIGURE 21. A screenshot of new table allocation after redefinition

The effect of these last three re-definitions (of the `seating_arrangement` and of the labels) could have been achieved by suitable dependencies, but at present these changes are typical of modifications by way of experiment until it is recognised by the user that they are reliably achieving what is regularly needed. Then they can be automated (for example by a 'combine tables' window).

On p. C-6 there is an example of an 'action' in tkeden:

```
proc customers_whohavebooked__entering : clock
{
      auto i,j;
       for (i=1;
       i<=booking_list#; i++)
    {

if ((booking_list[i][4]==clock)&&(booking_list[i][5]==FALSE))
                  /* i.e time to enter and not cancelled */
       {
          append customers_in_restaurant,booking_list[i];
           for (j=1; j<=booking_list[i][3]; j++)
          {
   occupyseats(seating_arrangement[booking_list[i][2]][j]);
           }
       }
    }
}
```

This is a procedure triggered whenever the value of 'clock' changes. It is a C style procedural body consisting of a standard pair of nested loops checking each tick of the clock to see if it is time for customers who have booked and not cancelled to enter the restaurant and be seated. The automation here is a simplification and one not really in the spirit of EM. Typically customers do not arrive precisely on time and all take their seats tidily as this procedure ensures. But where there are thoroughly reliable sequences of events such procedural automation (perhaps after some experimentation to gain confidence in it being appropriate) can be used to model it. The definition of `clock_string` for display in the clock window, for example, in the procedure also on p.C-5 would be a good example. To model more typical customer behaviour in an EM style we might have human users in the role of customers on other workstations in a distributed environment performing the actions of arriving at the restaurant, talking with friends, having a drink at the bar and eventually sitting down. This is the kind of modelling that is possible already with the tool `dtkeden` – in a schematic form sufficient for the interactions and feedback that are required.

Returning to the way this model might be used for table allocation, the visualisation at present shows an expected `booking_length` that is represented by a rectangle covering 90 minutes on the timetable. Currently this is initialised for all customers on p.C-1. If there is a base of regular customers a manager might well be able to customise the `booking_length` for known customers and therefore obtain a better prediction for the profile of occupancy during an evening. If someone telephones about 8pm in the situation shown in Figure 8 and wants a table for eight people at 9pm the decision depends crucially on local knowledge. For example, if only two of Wong's party turned up and they seem in a hurry there is a good chance of putting tables 5 and 6 together in reasonable time. There may be the physical possibility of placing table 1 with table 5, and, of course, it is possible that tables 6, 7 and 8 could be put together by soon after 9pm. Let us suppose Ward's `booking_length` had been customised to 75 minutes. Aided by this visualisation and her local knowledge of the parties concerned and (depending on the exact time of the request) her knowledge of the progress of the meals so far, the manager may then make the qualitative judgement that it is likely she can accommodate the party of 8 with little or no inconvenience (to

customers at least). In this way the visualisation and customisation support experiential knowledge and qualitative aspects of a situation.

There are a number of unexpected events that might take place when running a restaurant. There might be urgent plumbing, or electrical work, to be done making some tables suddenly unusable. Someone who had a won a lot of money might wish to buy drinks for everyone in the restaurant - thus suddenly interrupting the work of the waiters. Most of such unexpected events are unlikely to be taken into account in a traditional developed system. The RMM developed by EM is flexible enough to respond to arbitrary experimentation as well as responding to uncircumscribed events. For example, RMM enables arbitrary interactions e.g. various partitions could be inserted between tables, or groups of tables, to create a more homely or private atmosphere - we could experiment to see how much seating space might be lost by this action. This kind of redefinition 'on the fly' could not easily be achieved by using the traditional modelling tools whose possible interactions are fixed and which may not allow this kind of interaction. Having a chance to interact in this way, restaurant managers could reveal their experiential knowledge of the field, for example, in a certain area where an interior designer would not recommend having a table, for example, a door-way seat may appeal to a certain group of people. Knowing this, a restaurant manager could make a rearrangement and utilise the space. This kind of experiential knowledge may not be made public as there is so much that one knows and may not realise until the situations arises. The EM approach can also help with those difficulties mentioned above because experimentation leads to different degrees of certainty about knowledge and therefore supports incompleteness of knowledge and uncertainty.

In § 4.3 the claim is made that an EM environment can integrate the representation of propositional, experiential and tacit knowledge. One kind of tacit knowledge is 'skill knowledge', or knowledge of 'how to do' things. That someone has such knowledge emerges when, in the standard examples, they start swimming or riding a bicycle. It is the success of the performance that shows the knowledge. The way that the RMM can 'exhibit' the tacit knowledge of an experienced manager is through the 'his-

tory' record of the model. In any model it is always possible to view the trace of all interactions and re-definitions, including mouse movements for example, through the 'View History' option on the View menu. One could analyse through such a record those interactions performed by a skilled manager in dealing with a complicated series of bookings and cancellations. For example, consider the use of local knowledge for accepting a booking at 9.30pm for 10 people when the predicted availability shows only one spare table up until 10.15pm that can accommodate up to a maximum of 5 people. An analysis of such an interaction performed by a skilled manager, may reveal that a skilled manager has checked the arrival time of one of the earlier bookings. In this case, there is a group of 5 that arrived earlier than expected, and they were expected to finish soon. In a worst case, the party of 10 people may have to wait for half an hour longer to be accommodated at the restaurant which is a reasonable practice for late booking.

It is reasonable to assume such arrival times were logged by our computer system and incorporated into the model of the evening so there would be definitions representing such relevant propositional knowledge. But there would inevitably also be knowledge the manager might use that was not in the model. For example, she might know some of the party of 10 personally and be confident they will not mind waiting, or she might know that some of them are usually late in arrival anyway and little inconvenience will be caused. Such knowledge is not tacit knowledge, it is experiential. The tacit knowledge arises in the way all these other sources are treated in terms of interactive behaviours. The history trace would not always be an adequate *explanation* of the behaviour, but the behaviour would be an embodiment of the tacit knowledge of how to deal most effectively with the requests arising. Other managers could learn from this, and to some extent there can then be tacit knowledge being communicated through experience of the model without that knowledge ever becoming explicit.

Finally, one of the meanings of 'open-ended' that distinguishes an EM model from others is not only the 'unboundedness' of the interactions possible, but also the fact that the current interpretation of the whole model can be changed. For example, an easy re-interpretation of the RMM might be a seating plan organiser for private parties – e.g. a wedding reception – when what will matter most will be keeping track of combinations and proximities of relatives and friends of the bride and the groom. The timetable facilities then are of lesser importance as they apply to most of the party although there may be some variations. A greater change in interpretation (and therefore in the expected interactions) would be to regard the 'tables' as benches holding different experiments to be performed by a series of groups of students who need to visit them all in such a way that they minimise wasted time – that is benches not being used because groups have already done a particular experiment. In a conventionally programmed model such changes in interpretations, and consequent changes in possible actions, would normally be very difficult to achieve. But such 're-thinking' of a resource, or a business process, is not at all unusual in everyday life. The RMM is considered further in relation to strategic DSS in § 5.5.2.

5.4.2 DSS for unstructured problems

The application of EM offers potentially powerful alternative tools for Business Decision Modelling in the area of unstructured problems which will go beyond the capability of today's spreadsheet. Evidence from existing models suggests that EM as a generalization of the spreadsheet could become a fruitful approach in solving unstructured problems in the same way that the spreadsheet has been successful in the area of well-structured problems. The following excerpt from Sun and Beynon in [Sun+99a] supports the view that EM could contribute significantly in business modelling:

> *EM is a means of constructing knowledge in an experiential rather than a declarative fashion; the modeller's insight is expressed as coherence between expectations in the mind and the experiments that can be performed on the computer-based artefact and/or in the external environment. The principle resembles "what if" experiments with a spreadsheet. The modeller introduces new definitions*

> *to impose a change of state upon the embodied construal. Almost simultaneously, the new state of this construal is mediated to the user through the visual interface, and evokes a change of state in the mind of the modeller ...*
>
> [Sun+99a]

Earlier research work done at the University of Warwick has proposed that EM is an environment suitable for so-called creative software development, similar to product design development [Ness97]. The discussion within this thesis suggests another area in which EM might be suitable. The use of EM principles can be viewed as an environment for interactive system development, especially for an unclear domain, where the product is a system prototype. EM employs an experiential representation of knowledge via artefacts. The EM approach introduces the concept of transforming a domain situation into artefacts. Unlike a domain situation described in abstracted form of mathematics, a domain situation described by an artefact provides a more natural way of interaction between humans and the domain under study.

To make the human-system interaction more natural, an artefact built in an EM model will represent the domain in a pattern familiar to human cognition e.g. an individualized sketch drawing of any element within the domain under study may make more sense to the user trying to observe the behaviour of the domain than its representation by a set of mathematical equations. Nevertheless, the underlying representation of an artefact within the limited capacity of computer systems is still likely to be by mathematical formulae. The way in which the domain is represented as an artefact allows a real-world experience for the user or modeller in any interaction with the system.

Unlike EM, spreadsheet systems and some other traditional systems need discrete representations of values for their variables, that is, the modeller has to be specific and assign particular values for the variables. For example, in testing the various possible combination of carton sizes to use the capacity of a container fully via the implementation of a spreadsheet system, the modeller has to assign a specific value for each carton and then observe the effect via animation. The modeller has to think how much

space each different size carton will take up in the container before being able to observe the animation, so doubling the work load. However, interacting with the artefacts in EM, by using the mouse to control the artefacts gives a sense of immediacy to the interaction to the user – as if they were interacting with a real-world referent – not with the computer script or instruction set. For example, in finding how cartons of mixed sizes should be fitted in a container space, in an EM model the modeller can just drag the mouse to extend or shorten each carton, the model will then be re-drawn on the spot because of the dependency maintenance in the EM environment. It is not only the employment of dependency concept in EM that offers a better quality of interaction, but also another two major principles employed by EM: observables and agency.

The concept of observable in EM contrasts with the availability of only fixed discrete values assigned to any variable in a traditional system development which obstructs the natural interaction of modellers or users with the system. In EM any single possible observable in the system is a valuable element in the human activity of learning through experimentation, or in other words, gaining 'experience'. New observables can be monitored through an interaction with the model and they may lead the modeller to a new understanding, either a totally new idea or a developed insight. The second concept of dependency works similarly to the spreadsheet. The on-going research work aimed at developing a higher order dependency maintenance is another step in offering a more powerful tool for dependency handling as compared to a spreadsheet system. The agency concept in EM introduces a new possibility for model development that can be applied to the concept of concurrent engineering via the use of agents. The advantage of this concept is that it allows the development of a complex system like those in business practice where one thing seems to have connection with many others and where it is hard to pre-define the order of priority among them. This is one reason why the traditional approach in developing a business system has had limited success. The concurrent structure within EM allows the building of artefact 'Z' to be developed before the construction of artefacts 'A', 'B', 'C'. These

concepts are well suited for ill-defined problems in which no clear preconceived view of the possible actions could be described.

5.4.3 Extension of mental models

On the assumption that the graphical image is a basic representation, that is, a natural form of human cognition, the use of EM's artefacts (a visual representation of each element) to represent or model the domain under study helps the modeller or user to develop insight or to better understand the domain. The development of an EM model is intended to show that the explicit representation of the state of the mental-model of a modeller in a particular area under consideration assists the modeller to focus on critical thinking rather than imaginative thinking (thinking back and forth to recover their judgement of particular factors that have been considered earlier since the memory can hold only a limited number of factors at one time). This not only helps in recording what they might later forget, but also helps in representing and revealing users' current thoughts toward each particular factor. If this is not explicitly shown, users might not be aware of their particular attitude which might have a serious effect on their judgement or interpretation of the situation.

In Figure 22 shows a simple design of an ISM model that a manager might create to help monitor their qualitative judgement about a particular market. The drawing of a house represents a target market. The idea is that the more cells in the house that are painted the better is the potential of such a market. The columns represent a particular factor, the height painted of each column indicates the degree of that factor in the situation in the judgement of the user. The width of the column indicates the weight the user wishes to give to that factor. The lower screenshot is an EM model of classroom interaction which has been used to simulate student behaviour in relation to how the interaction with their classmates and a teacher could result in particular decisions of how to behave. There is a potential reuse of this model in a market simulation model for a marketing manager to see what he or she has in mind about the situation of the market and how he or she can experiment with it. In such reuse, each student face can represent a factor of the market, e.g. customer behaviour, government policy,

stakeholder attitude and staff satisfaction. These factors are all based on the personal interpretation of the information gained from quantitative analysis and personal observation. The quality of this kind of interpretation varies from person to person and may make a critical difference in making a decision.

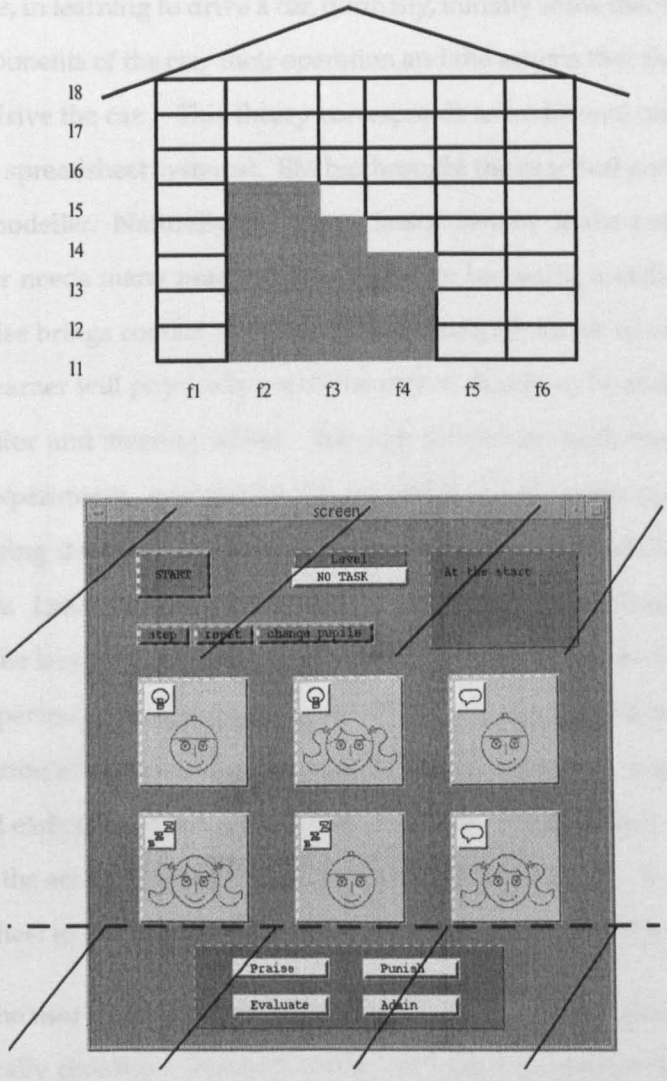


FIGURE 22. A personal mental model of a market

EM offers a richer visual representation than the spreadsheet system. The models developed by an EM approach have more in common with human cognition in terms of the richness of its data types, especially the facility for the user to specify any type

of data to suit individual applications. This is not only because of its methodology in introducing a natural way of interaction between human beings and the model being developed via the artefacts, but also the underlying principles governing observables, dependencies and agents offer a natural means for experiment, experience, learning and re-learning any subject matter.

For example, in learning to drive a car, normally, initially some theory will be given about the components of the car, their operation and the actions that the learner needs to perform to drive the car. This theory corresponds to traditional computer modelling (including spreadsheet systems). EM has brought the practical part of the driving lesson to the modeller. Naturally, the theory lesson cannot make a skilled driver of itself. A learner needs many practical lessons before becoming a skilled driver. The practical exercise brings contact with reality to the subject matter of car driving. For example, the learner will physically touch the objects that have been described as the brake, accelerator and steering wheel. Through interaction with these objects, the learner then experiments, experiences, learns and re-learns those matters that have been given during the theory lesson and develops the necessary skill to operate and monitor the car. Later on, through the continuing process of experiment, experience, and learning, the learner gains insights into the subject matter and becomes capable of developing expertise so as to become an expert driver. It is not simply that EM supports a simulation of the driving experience but the environment supports arbitrary extensions and elaboration of experience. For example, a learner may try to press the pedals of both the accelerator and the brake together at once and see the effect, or turn the steering wheel to one extreme and keep it there while pressing the accelerator.

Likewise, the user of any software, e.g. word processor or spreadsheet systems, has to have physically consistent interaction with the software before becoming a skilled user. In the case of software users, after the theory lesson, the learners do not have direct contact with the program instructions any more than learner-drivers have contact with the engine specifications. The keyboard, mouse, touch screen and voice activator are examples of the currently available media of human-computer interaction, with which the operators must become skilled, as the learner driver with the brake,

accelerator and steering wheel. A proper interface is then a crucial factor in any interaction system to be useful for business people. E. Turban states, *"An inconvenient user interface is one of the major reasons why managers have not used computers and quantitative analyses to the extent that these technologies have been available ..."* [Turban95, p. 99]

"A DSS is more a service than a product. Since the problem can only partially be structured and since managers grow in their understanding and needs over time, a DSS must constantly grow and evolve as the user adapts and learns ..." [Turban95]. This reflects how the method and principles of EM could lead to a new direction for a practical DSS development. A speculative explanation of which area of decision support need that the EM approach could serve will be given in more detail in the following sections.

5.5 Strategic Decision Support

Decision making is a central part of business practice at every level of management. Sutherland [Sutherland98] classifies three levels of decision making: strategic, tactical and operational. The strategic level has to do with formulating and choosing among alternatives that are different in kind ('qualitatively disparate'). A decision support system is one in which the human user plays an essential, interactive role. This makes the integration, and the qualitative nature, of an EM approach particularly suited to a strategic decision support system.

Strategic decision support systems (SDSS) present three particular additional challenges of their own: (i) strategic problems are typically imprecisely described and qualitative; (ii) strategic decisions are likely to be taken by the most senior managers and so ease of use, or the possibility of end-user development, is highly desirable; (iii) strategic decisions are likely to be shared among managers and so require a distributed system (i.e. a group DSS).

In the following sections we examine how a current conventional tool explicitly designed for SDSS compares with the very different way in which an EM environment can address the issues mentioned above.

5.5.1 The use of DE to support strategic decision

Decision Explorer (DE) is a computer-based support system for an approach in strategic decision: SODA. SODA stands for Strategic Option Development and Analysis. It is an approach initiated by Professor Colin Eden and Frank Ackerman in performing strategic management activities. Professor Eden and his collaborators initially at Bath University and later on at University of Strathclyde have also developed software initially called COPE. Decision Explorer (Cognitive Mapping) is a commercial product derived from COPE (as early academic version of DE) which is distributed by Banxia Software Ltd. in UK.[1]

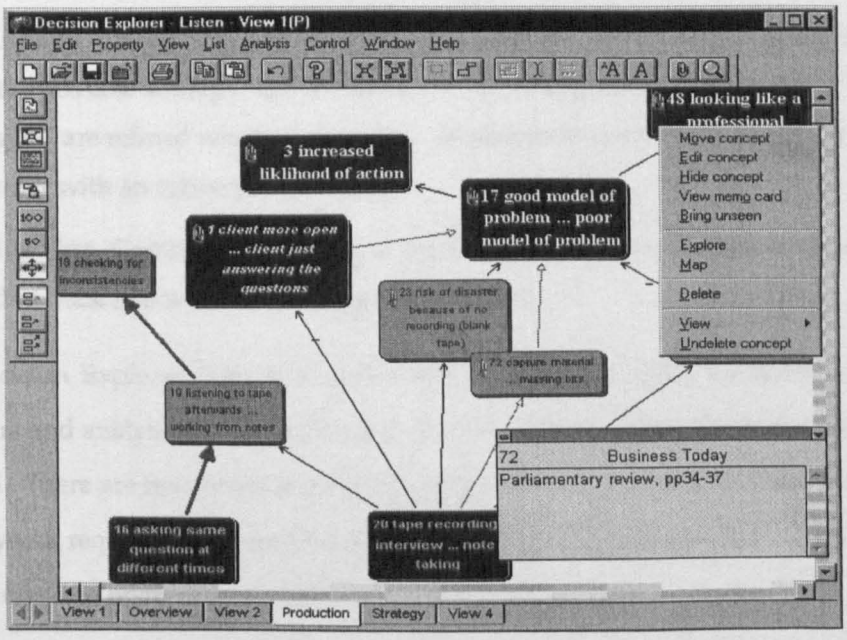


FIGURE 23. Screenshot of Decision Explorer (http://www.banxia.com)

SODA is based on the technique called 'Cognitive Mapping' which is founded on George Kelly's theory of personal constructs – a predict and control view of problem solving. SODA is a methodology that uses cognitive mapping to express the thoughts

1. Discussions and materials used in this section are based on the information obtained from http://www.banxia.com

and opinions of a decision making group to support strategic thinking. The SODA process, in general, is a combination of the following steps.

1. Interviews: this step refers to a relatively open and unstructured discussion to identify key issues in the interviewee's thinking. This step is also custom made to suit the interview agenda. Questions are built on one another and further questions are revealed by a cognitive map.

2. Computer Modelling and Analysis: this step refers to the construction of a model of knowledge and argument to explore the different opinions in order to identify goals, supporting assumptions, key issues, options and actions.

3. Group Workshops: this step leads to the process of elaborating the model and focusing on emergent goals.

4. Group Decision Support Workshop: this step leads to sharing and identifying the possible strategic options and the number of goals of each option. Then the goals are refined where appropriate. Strategies to meet the goals are agreed together with an action programme.

5. Tracking, Control and Review: a model of the SODA process can also be used to track and monitor the progress of actions.

Decision Explorer (DE) is a system software that provides an environment to structure and analyse the qualitative information circumscribing the issues being considered. There are two forms of information that DE can deal with: i) straight forward ideas which require further exploration and scrutiny; ii) complex ideas which require structuring and analysing to cope with the complexity.

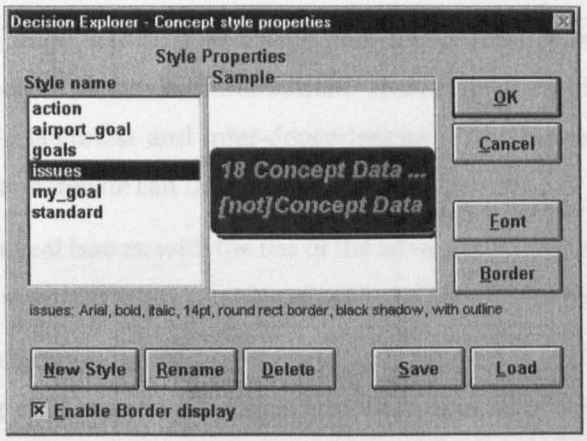


FIGURE 24. Screenshot of choices of concept properties in DE (http://www.banxia.com)

DE is an environment for modelling which employs a mapping technique to interlink ideas. A map of ideas is referred to as a model which is a representation of a data set as a whole. DE's map is made up of short phrases or concepts and the links which show the relationships between the concepts at each end of the link. Both concepts and links are established by the users. The map type that is employed by DE can be referred to as a 'cognitive map', a 'cause map' or a 'concept map'. Decision Explorer can be used:

i) for any situation that assists the structuring and analysing of qualitative information;

ii) to understand and arrive at conclusions about issues involving complexity and/or uncertainty;

iii) as an individual or group decision support tool;

iv) as a communication medium in exchanging of ideas;

v) for mapping interviews and

vi) as a tool to demonstrate the chain of events or exercise project management.

The key benefits from using DE are:

1. **To improve understanding of a situation:** the process of mapping ideas promotes a coherent picture of the situation. Then an explicit picture helps to show the inter-relatedness and inter-dependencies of the issues about which an exploration and debate can take place.

2. **Discover the real issues:** with the use of the advanced analysis facilities provided by the system, the users can identify the real issues of the situations.

3. **Resolve conflicts and provide more practical, acceptable and feasible solutions:** by being able to consider different options and opinions, this activity leads to expanding argument and reasoning in order to arrive at a common agreement within the team. Then, when this common agreement is shared it leads to more acceptable practice for everyone in the team.

4. **Maintain richness of data**

5. **Help store the current state of the issues being considered:** with the use of DE, the current state of the issues being discussed are kept in the computer memory and available for further discussion if there is any interruption during the discussion. This reduces the time spent in referring back to previous ideas under discussion and focuses attention on the issues rather than repeating old discussion.

Cognitive mapping is a prominent technique developed in psychology that has been used in business processing and forming strategic choices to support the decision making process of top management. With the use of the computerised system decision explorer, the technique seems to be useful also in some other business processes, e.g. stakeholder analysis, corporate memory and process flow control.

Stakeholder analysis aims to identify the organisation or individuals who have an interest in and can influence a business. The mapping is used in this activity to gain a clearer and more integrated picture of the relationships between the organisation and its stakeholders. The use of the map will also highlight areas which can be exploited through some form of collective agreement and areas of potential difficulty and conflict. Mapping is also used to express and show the stakeholders' characteristics (their

perceived goals) and then to relate the company's own goals to each individual stakeholder. The inter-relationships are revealed by two questions: i) how does the stakeholder contribute to the achievement of organisation's goals; ii) how does the organisation contribute to the stakeholder achieving their goals.

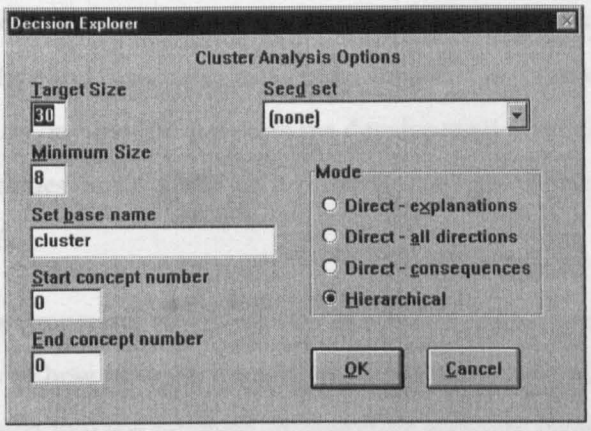


FIGURE 25. Screenshot of options of analysis in DE

Corporate memory can be achieved by the use of cognitive mapping. A map provides a convenient and efficient way of storing information. A map can be used to provide corporate memory in two ways: i) to record the reasoning behind decisions, not only the decision outcomes. The mapping process can help capture the thought process in meetings and allows a revisit when required; ii) to record external strategically significant information. Event maps, which are maps that show chains of past, current and possible future events occurring in the organisation, can be circulated and periodically updated. This kind of map can be used to identify trends in the business environment.

Process flow charts are enabled by mapping to facilitate the Soft System Methodology (SSM) process. SSM offers a way for the member of an organisation to define primary tasks. Primary tasks are those transformations that the members of an organisation think that their organisational system should be able to achieve or perform. For example, customer demand leads to profit. Library service leads to an enhanced provision for local education. Decision explorer provides a high level of

data flow diagrams of a set of interconnected activities derived from analysing the transformations.

5.5.2 The use of EM to support strategic decision

The RMM (see § 5.4.1) is chosen as a case study for an implementation of an EM approach to DSS development. This is because it provides a compact environment and both quantitative and qualitative aspects are included and can be illustrated. The following discussion on how EM supports the development of strategic decision support systems continues using RMM as a running example of properties of EM for SDSS as well as DSS.

EM offers an environment, where the way in which a system is developed imitates the characteristics of how humans naturally perform a certain activity. For instance, consider the decision making involved in allocating tables to customers in a restaurant. At present the RMM makes an automatic suggestion for table allocation following the obvious algorithm of selecting the first table of adequate size that is available at the time of the booking. In complex situations this could not be automated for optimum occupancy and so we would use the system to support the manager's own decision making. There are a number of observables and dependencies among observable elements and both controllable and uncontrollable agencies which affect the state of observables that will need to be taken into account. The total number of bookings for a particular night, the characteristics of customers e.g. whether a group party, a couple or a non-smoker, the customer's preferences for a type of table and the time of arrival or reservation are examples of observable elements. These observables have mainly qualitative values and are subjective. This type of data is found to be best manipulated by the human brain based on previous interaction and experience. This type of data is hard (if not totally impossible) to represent by any direct quantitative or mathematical model(s) which are the only available kind of models for current management science. Therefore the principles of the Empirical Modelling approach introduce a richer type of data set to suit any application and the distinct 'quality of interaction' brings a new direction for management science.

In deciding a table-allocation for a group party, there are a number of dependencies that matter. For example, the total number of persons, the available space to join individual tables into a party table (which depends on the size of the group and the availability of individual tables and their locations). These variables are observables which naturally depend upon others. There is a controllable factor that governs this decision situation which in this case is the manager's assignment of a particular table for a particular party. There is also an uncontrollable factor, e.g. a customer's strong and instant request for a specific table which may affect the pre-planned arrangement and lead to a new decision. The observables, the dependencies of one observable upon the others and the agents (e.g. a manager who is considered as a controllable agent, and a customer who is considered as an uncontrollable agent) that are responsible for any changes, are major components of the system that represent the state of affairs in the EM approach. This aims at being a computer-based model which reflects the view of a human agent (e.g. managers) and enables observation and experimentation to reflect the uncontrollable agent, which may affect the system one way or the other.

In constructing a restaurant management model (RMM), we have considered that the EM environment is a suitable environment for the development of a system, in particular, the strategic DSS [Rasmequan+00a]. As discussed earlier, any DSS aims at supporting the generation and analysis of alternative solutions to a problem, and EM's 'what if' feature is well suited for this purpose. It should be easy to imagine now how the combination of semi-automated booking decisions with locally informed choices by the manager could help explore consequences for the later bookings of alternatives being considered for earlier bookings. Some of the crucial elements for a strategic DSS are i) the capacity to represent imprecise and qualitative aspects of a situation, ii) the need for end-user development, and iii) the value of distributing the model to several managers. Each of these will now be considered in turn.

Some scenarios described in § 5.4.1 are about booking requests for large party at times when 'strictly' tables of sufficient size are not available. But it was clear from discussion there that the practical and cognitive issues are imprecise and qualitative.

(For example, how long are the customers willing to wait ? Would the party be willing to be seated in two groups ? Can two particular tables be put together without inconvenience to other customers ?) It is because of the degree of integration of human processes (qualitative judgements) and computer processes (the display and storage of background information and interactions) in an EM model that the qualitative aspects can be said to be represented by the RMM (see § 4.4.3). The judgements and the information, their context and the interactions are mutually dependent upon each other. So the relevant agents: the manager as modeller and user, the RMM as computer artefact, and the restaurant itself form a 'system' which is reflected within itself in the RMM.

The goal of end-user development is not yet a practical possibility; the tools being used need more technical knowledge than can be expected of a business user. But in principle this goal is realistic because of the nature of the modelling method. As already argued these models have a claim to being natural (§ 5.3.1) and to being extensions of mental models (§ 5.4.3).

The distributed modelling tool, *dtkeden*, has been described in § 3.2.2. This allows several users to collaborate in interacting with the same model and permits variety of modes of communication. Thus several managers could co-operate in strategic decision making use of a common computer artefact. Quite complex multi-user distributed models of a railway accident and 5-a-side football involving 6 - 8 workstations have been built in recent years.

An EM model enables the process of strategic decision to be explicitly traced. Normally, this process perhaps is implicitly constructed in a manager's head. For example, a restaurant manager may unconsciously construct a strategic decision model while managing a seating allocation on a busy night. A restaurant manager's strategic decision model might, for instance, comprise three major components, as follows. The first component is the concern for profit maximisation which depends roughly on the number of the customers, the allocation of seating (maximized space) and the timing allocation (2 -3 slots if possible for a particular table for that particular night). In this

particular area, there is an analysis of estimated takings based on the number of the customers and the possible arrangement of the seating allocation. This type of analysis as done by a human agent without computer support lacks consistency in keeping in mind the current state which is going to be a crucial element in further analysis. The second component is the concern for customer satisfaction which roughly depends on the quality of the food, quality of service and the atmosphere, taken together with the cost. This concern is based on the manager's personal observation of the reactions of customers. This is purely experiential knowledge. If this type of knowledge could be captured and represented by a graphical representation (one of the most suitable modes of representation for human cognition), as one of the elements of the model, then this type of model might be useful as an expert knowledge-based system in a particular domain. The third component is the concern for staff morale and performance which depends on the number of customers, the customers' requirements, the seating allocation and the atmosphere. This component is of similar quality to the previous components and is a quite predictable qualitative element based also on the manager's experience. The elements in each of these components are linked together in ways that, in some cases, are complicated and require experience and experiment to determine.

5.5.3 Comparison of DE with EM modelling

The DE tool described in § 5.5.1 is typical of many business applications. It is fundamentally a tool for flexibly storing, manipulating and displaying text. A DE map consists of "short phrases and concepts" with links representing relationships between them. The analysis of DE as shown above indicates that DE has been used to support strategic decision making to help accommodate the process of recording and displaying the ideas of executives by way of manual strategic analysis (SODA). This has also been used – in a spirit consistent with our thesis – to help identify a problem or to understand better the business situation. In addition, it can be used to be an extension of the mental model of a manager in making a qualitative judgement.

The RMM, as an example of EM modelling, displays an entirely different quality of model: namely a model which is not merely textual or conceptual but one which can give rise to experience. Even though that experience may be given via very simple graphics and metaphors these can be sufficient to reflect faithfully the patterns of interactions and observables needed for the representation of sophisticated knowledge.

Chapter 6

Conclusion and Future Work

6.1 Overview

This research work has investigated how to apply an alternative approach to computing, Empirical Modelling, to the use of computer-based support systems for the development of business solutions. The major contribution of the thesis has been in bringing issues and ideas from the field of knowledge representation into relationship with EM principles and methods on the one hand, and the needs of business managers for support systems, and DSS in particular, on the other hand. The study has confirmed that there is substantial potential for the application of EM environments to business support and that the study of cognitive processes offers many fruitful links between EM resources and business requirements.

The review of business support tools in Chapter 2 emphasised that the progress over the past two decades in the DSS that offer significant cognitive support has been very slow compared with the steady improvements to functional tools like accounting packages and databases. There has also been little progress with tools to take account of qualitative aspects of situations, or of non-propositional knowledge, or to allow serious end-user development.

Empirical Modelling was introduced in Chapter 3 as an alternative approach to computing. Some fundamental issues central to EM such as state, experience and abstraction, were discussed from a business perspective and with a view to how far the EM approach could address the limitations identified in Chapter 2 in conventional support tools. While the technical details of EM presented there were not original, the discussion of the key concepts and motives in relation to business needs is original.

Two original contributions of the thesis appear in Chapter 4: i) the identification and explanation of how the methods and tools of EM, and the quality of interaction in EM, serve to provide a new form of knowledge representation (§ 4.3); ii) the understanding and explanation of how EM artefacts serve not only to integrate propositional, experiential and tacit knowledge, but more generally how the EM approach serves a powerful integrative function for computing (§ 4.4.3).

Chapter 5 contains a detailed illustration of some of the explanations and claims about an EM environment and knowledge from Chapter 4 (§ 5.4.1 and § 5.5.2). It contains original arguments for why EM is a 'natural' environment (§ 5.3.1), for how ISMs are well suited to serving business needs (§ 5.3.3) and how EM methods already address in practical terms some key elements of strategic DSS (§ 5.5.2).

6.2 Assessment

This research area was shaped by personal interest in seeking the answer to the following question: "Why isn't there any computer-based support system that is flexible enough to handle my requirements, in particular, on what counts and makes sense to me (as a non-programming user) in dealing with business problems ?". Spreadsheet systems are the only kind of system (in my experience) where I feel I have some control over the system to initiate my thought and help me to reflect my domain knowledge of a particular business practice. However, there are some constraints when working with spreadsheet systems. They are the limitations of its functionality and flexibility. In addition, spreadsheet systems are more likely to work well at the level of personal support systems, rather than systems on an organisational scale.

Such difficulties with the use of conventional computer-based support systems urged me to pay attention to any novel area of computing that might offer an alternative choice. The principles of the Empirical Modelling approach have been found to be interesting and have been explored to see whether this might be the right kind of environment that could give an answer to such questions. During the years of this

research work, there were a number of unexpected difficulties and benefits that shaped the work to its current stage as follows.

As the accounts of current knowledge-based systems and of DSS in Chapter 2 have shown, there are an enormous number of pragmatic techniques and tools being used in business. All kinds of techniques are being used whenever they might be useful. Not only is there a lack of theory for business problems, or for the management and use of information, there is a lack of any principles on which to base the application and use of technologies in business. EM brings concepts and principles which people use in making sense of their personal world, and the world they share. These principles are broad enough to evaluate the cognitive processes that people use all the time in business, for example, the need to do detailed interaction with a project management team while simultaneously interpreting and evaluating the progress and prospects of their project. But the concepts and tools of EM also support detailed practical work allowing observation-oriented, incremental change to a model to be used at all times during the model building process.

Dealing with subjective kinds of concepts, such as experience-based, interaction, cognitive artefacts and knowledge representation causes this research to search for a deeper understanding to be able to argue and discuss concretely about them. A great effort has been made to collect and understand extensive but related materials in philosophy, psychology, computing, information systems and business studies. The language barrier also adds further to the effort to appreciate them in the first place. However, this effort yields a reasonable outcome. Nevertheless, there is further scope to demonstrate these findings by practical work. The limitation of the time scale of this research and the limitation of the tools to support such demonstration have not permitted this. Further demonstration by practical work could help reduce the speculative quality of this thesis.

A conclusion that can be made from this research indicates that an EM environment offers a number of distinct properties that are different from those offered by a conventional environment. Such properties are:

- Giving computer-based knowledge representation. This means EM models support a combination of kinds of knowledge. Such kinds of knowledge are:

  propositional knowledge, *for example, those physical laws such as* $E = mc^2$*, or a financial formula such as Interest = Principal Amount × Interest Rate, embedded in particular components of the models;*

  experiential knowledge, *The observation and interactive interaction with the simulation model of a market analysis model using spreadsheet like notation (definitive scripts) can offer a similar kind of experiential knowledge to that which develops within a person when he or she interacts with a real market situation;*

  tacit knowledge, *for example, we know how to perform a certain task e.g. ride a bicycle, or play a badminton game, but we are hardly able to explain clearly or verbally how we can carry out such tasks. This is very similar to how a person conducts herself in business activities, e.g. a sales manager might naturally (based on his or her experience) be able to configure, integrate or analyse a certain number of factors in his or her business model when deciding his or her business strategy. This sort of thing is why the manager sometimes finds it difficult to explain why he or she has made such and such a decision.*

- The modelling process of EM emphasizes the use of cognitive artefacts as a medium to communicate and facilitate the thought processes of the developer herself, or of a group of domain experts. This can be referred to as cognitive modelling which, to some extent, is similar to cognitive engineering work for people in the designing area.

- There is a flexible degree of integration between humans and computer systems which enables the integration between *hard & soft data* and *hard & soft components* that are very critical factors in conducting any purposeful compu-

ter-based support systems for business activities. This is very much concerned with the nature of a business environment which is human-centred.

These are some major properties that enable EM to offer a good environment for developing effective business solutions.

The contribution of this thesis is mainly based on the recognition of the necessity of an alternative approach for computer-based support systems for the business environment which has an ultimate goal towards user-developed systems. In response to such recognition, the *Empirical Modelling* approach is identified as one of the approaches that offers a very promising new paradigm with substantial evidence of both theoretical and practical significance for business applications. Some demonstrations are included to show how EM offers a proper mixture of components that enables the building of improved and more effective human-centred computer-based support systems.

6.3 Future Work

The developing of business models to exhibit the theoretical arguments made in this thesis about a special form of knowledge representation is a future work that can immediately build upon this thesis. Such developments could demonstrate how the nature of EM environments enable better support for unstructured problems of business people.

Another possible direct link to future work with this research is the application of a more user friendly environment to build an actual decision-support system for a business firm in action. In the initial state of development, this work would need close cooperation between a company interested in using an EM approach for its business development and an EM expert.

The Soft System Methodology (SSM) approach, which is based on the acquisition of knowledge through a learning process, has a number of principles in common with EM's principles. We believe they can be combined with the EM practical tools and we

wish to propose a framework in which they can be combined. In this preliminary study, the aim will be to find whether the SSM approach would be able to complement the EM approach in developing such an environment as they share some major principles. SSM has been selected to be a supportive approach as it claims to have the ability to improve the original problem situation. [Checkland+90]. Among a number of difficulties in systems development, the understanding of the problem situation of systems being modelled is a major one. The proposed research will be conducted by exploring whether EM environment and tools can be used as a means of implementing analysis carried out using a combination of principles from SSM and EM. Then some evaluation will be conducted via a field test (this might take the form of a co-project with business school to test the constructed environment for building up Business DSS prototype by business students). The following is a result of the preliminary study of the relationship of SSM with EM.

Empirical modelling as a computer-based instrument for Soft System Methodology

SSM and its principles. Soft Systems Methodology (SSM) was developed in 1970 by Peter Checkland, in his response to the problem of managing situations and people in the work place. This methodology is proposed as an organised way of tackling complicated situations in the real world. Being founded on systems thinking, it is highly structured but flexible and broad in scope [Checkland+90].

P. Checkland and S. Holwell [Checkland+98] point out that SSM is usually concerned with ill-structured problem situations, such as what to do about researching new products given competitors' innovations. The main concern here is to deal with social reality in human groups which are continuously socially created in a never-ending social process. This nature of social reality is not like the physical regularities of the universe which natural science studies. The wave of never-ending social processes brings changes through time. This methodology is aimed at giving explanations via dynamic models which can cover both phenomena of social reality, that is persistence of human institutions, and changes through time.

The underlying concept in building a 'human activity system' is based on the idea that there is a situation which at least one person sees as problematic. Such situations, in the view of SSM, involve people who try to take purposeful action. This 'human activity system' model is not meant to describe such situations but to explore them coherently. A 'human activity system' model consists of two sub-systems:

> i) *a set of activities linked together according to their dependent relationships so that the whole would be purposeful; and*
>
> ii) *a monitoring and control sub-system so that the whole could in principle survive in a changing environment.*
>
> [Checkland+98]

Principles shared by SSM and EM. There are four major principles, shared by SSM and EM, that suit system development where the social context is regarded as of equal importance to the technical context e.g. DSS which have become complex multi-dimensional information systems. Those principles are:

1. **Applying the scientific method** – SSM which is rooted in Systems Engineering (SE) considers the system being modelled as an object. This is consistent with EM's concept of a system being modelled as an artefact. By this, EM refers to a system being developed in a manner similar to the development of a scientific instrument or engineering prototype.

2. **Consideration of the human factor** – Beynon [Beynon97b] argues that EM is grounded in the consideration that cognition and learning are fundamentally concerned with the process of construing phenomena in terms of agency and dependency. In addition, the multiple viewpoints of interpretation are also introduced under EM principles, as well as learning processes. According to Checkland and Scholes [Checkland+90], SSM is based on the 'human activity system' and considers that one of the most obvious characteristics of human beings is their readiness to attribute meaning to what they observe and experience. Checkland and Scholes point out that SSM is a methodology for operating the endless cycle from experience to purposeful action and back again.

3. **Concepts of experiencing and experimenting** – Both SSM and EM put the same high priority on humans' ability to learn or acquire knowledge through experimenting and experiencing.

4. **Suitability for ill-defined problems** – Both SSM and EM argue their suitability for ill-defined problem types. This is by way of modelling the problem or situation under study in a way that leads to understanding.

Properties of EM to Support Computer-based SSM. EM has an unusual programming philosophy which relies on observation and experiment. There are two principal techniques for analysis and representation in use: the first is the definitive representation of state and the second is the observation-oriented analysis of agents. The definition of the EM method has its basis in observation and experiment which is well suited to reactive systems and which provides an integrated environment for the requirement, specification and design phase of a development. EM applies the concept of agency in a reactive system by considering that agents within the system react differently according to the situation around them.

As stated in [Checkland+98, p. 155], Checkland *"... introduced soft systems methodology (SSM) as an interpretive approach to organizational problem solving which can be used to provide a structure for action research in which desirable change and organizational learning are the aims. Frequently, that change and learning is associated with the design, introduction and use of information systems..."*. EM has the potential to be used as a modelling process for designing and introducing information systems in response to what is required to be changed and what is supposed to be learned under the concept of SSM. This is because the EM environment promotes an unusual modelling process in that the process of constructing the model can lead to a better understanding of the problem domain. This is unlike a conventional modelling process that requests a clear problem specification.

Other work

Finally, there are a number of ongoing research studies connected with an EM approach. For example, the study of how to apply an EM approach to business process engineering compared to the use of OO techniques; the application of an EM approach to financial enterprises for the stock market; the development of user-friendly environments; the study of definitive notations in comparison with formal methods; technical work on the development of new unified and easier-to-use notations, and the study of how an EM approach can make a major contribution to learning and education. Each of these studies is easily related in one way or another to issues of knowledge representation. Consequently it is expected that the contributions of this thesis concerning knowledge in EM environments will relate to future research in many areas.

Appendix A

Constructing a simple clock model with definitive scripts

Note by Meurig Beynon

When creating a model by using a computer, the normal practice is to decide the precise functionality of the model in advance, and to implement from a functional specification. Modelling activity in EM is closer in spirit to creative work in the arts such as making a sculpture or composing a piece of music. The interaction between the artist's state of mind and the work they are creating is dynamic, and the meaning of the work of art is shaped as it is being developed. A simple exercise in modelling a clock can be used to illustrate this idea. The basis for this model is the file of definitions clock.d that comprises a definitive script in which the variables correspond to familiar observables associated with an analogue clock, such as the marks that indicate the time and the lengths and positions of the hour and minute hands. Features of this file include the dependencies that link the positions of the hour and minute hands to the current time. Notice how these are specified in such as way that both the position of the minute hand and the hour hand depend on the time via independent definitions. An alternative way to express this dependency that might more aptly describe the physical relationship between the hands of a mechanical clock would express the position of the minute hand as linked to the position of an internal mechanism, and derive the position of the hour hand by a definition representing the chain of cogs that might connect the hour hand to the minute hand. The file clock.d generates a clock face where the hands are yet to be displayed since the time (clock/t) is yet to be specified. In specifying a time for the clock face, the modeller can adopt many different viewpoints. For instance:

· the modeller may act as a user, setting the clock to the current time

· the modeller may act as a designer, seeking to place the hands in a significant configuration

· the modeller may act as the clockmaker who inserts the clock mechanism. Possible definitions to complement clock.d that correspond to each of these scenarios are given in ext0.d. There are many other instances of potential redefinitions in the file ext0.d. These effect very simple changes to the generated display, but ones that nevertheless can correspond to rich thought processes and changes of perspective on the part of the modeller. In the role of a user, the modeller will consider such issues as starting and stopping the clock, or setting the time to reflect a new time zone. In the role of designer, the modeller may consider the appearance of the clock face, the possibility of adding a second hand, or changing the colour of the hands. The modeller can also act in a role that is outside the scope of either the designer or the user, as when reconfiguring the display to a convenient size for demonstration, or adding physically unrealistic features to the clock. The motivation for making these modifications is to highlight two fundamental ideas behind Empirical Modelling:

1.the construction and structure of scripts mirrors the way in which the modeller construes state-change to occur

2.the modeller's perspective on the script is subject to change from moment to moment, and involves internal human activity (relating to thought processes, situation and agency) that is much richer and more complex than the external computer-based change.

Note how traditional computer programs, in contrast, being optimised to serve particular functions and operate in specific situations, are constructed in ways that do not necessarily give any insight into the fashion in which the programmer construes the world (though this is recognised to be highly relevant to the process of identifying a requirement). Moreover, they are generally intended to exploit the computer's capacity for performing exceedingly complex state-change, and to make the role of the user as clearly defined and as simple to enact as possible.

```
/*clock.d*/
%scout

integer clockwinScale;
point clockwinNW;
point clockwinSE;
point clockwinPos;

window clockwin = {
  type: DONALD
  box: [clockwinNW, clockwinSE]
  pict: "CLOCK"
  xmin: 30
  ymin: 30
  xmax: 370
  ymax: 370
  bgcolor: "white"
  border: 1
};

clockwinPos = {100, 100};
clockwinScale = 2;
point clockwinNW = clockwinPos + {20 * clockwinScale, 20 * clockwinScale};
point clockwinSE = clockwinPos + {180 * clockwinScale, 180 * clockwinScale};
display screen = <clockwin>;

%donald
viewport CLOCK

openshape clock

within clock {
  real sixthpi
  line eleven, ten, nine, eight, seven, six, five, four, three, two, one
  line noon
  point centre
  real radius
  circle edge

  sixthpi = 0.523599
  radius = 150.0
  eleven = rot(noon, centre, -11 * sixthpi)
  ten = rot(noon, centre, -10 * sixthpi)
  nine = rot(noon, centre, -9 * sixthpi)
  eight = rot(noon, centre, -8 * sixthpi)
  seven = rot(noon, centre, -7 * sixthpi)
  six = rot(noon, centre, -6 * sixthpi)
  five = rot(noon, centre, -5 * sixthpi)
  four = rot(noon, centre, -4 * sixthpi)
  three = rot(noon, centre, -3 * sixthpi)
  two = rot(noon, centre, -2 * sixthpi)
  one = rot(noon, centre, -sixthpi)
  noon = [centre+{0, 0.9*radius}, centre+{0,radius}]
  centre = {200, 200}
  edge = circle(centre, radius)

 line minHand, hourHand
 real minAngle, hourAngle
 int t
 minAngle = (pi div 2.0) - float (t mod 60) * (pi div 30.0)
```

```
 hourAngle = (pi div 2.0) - float (t mod 720) * (pi div 360.0)

 minHand = [centre + {0.75*radius @ minAngle}, centre]
 hourHand = [centre + {0.5*radius @ hourAngle}, centre]
 centre = {200, 200}

}

# clock/t is a donald int variable
# representing time elapsed in minutes
# from midnight (1440 per day)
```

/*autotime.e*/

```
/* This eden file illustrates the concept of automating a pattern
of observation. The underlying construal is:
The watch automatically consults the real-world time (via
the action new_time = gettime(); in chk_time), and updates
the number of seconds elapsed since 1/1/70 (recorded in tn)
accordingly.
Issues:
How fast does chk_time act and schedule actions (todo())?
"Accordingly" means "if the time recorded in new_time changes"?
How much real-time elapses between tn = gettime(); and the
initiation of the clock mechanism with first invocation of
inc_time();?
*/

tn = gettime();

/* use a day long absolute time for testing purposes */
func abs_time {
auto h,m,s;
h=$1[3]; m=$1[2]; s=$1[1];
return h*3600+m*60+s;
}

func ext_time {
    auto h,m,s;
h = $1/3600;
m = ($1%3600)/60;
s = $1%60;
return [s,m,h];
}

/* inc_time updates the time second by second */

proc inc_time {
auto tnt, tailtn;
tailtn = [tn[4],tn[5],tn[6],tn[7]];
tnt = abs_time(tn);
for (i=1; i<=$1; i++) {
tnt++; tn = ext_time(tnt)//tailtn;
}
}

old_time = gettime();
proc chk_time : new_time {
todo("new_time = gettime();");
```

```
if (old_time[1] != new_time[1])
{
inc_time(1);
old_time = new_time;
}
}

new_time = old_time;
/* need to initialise this in order to trigger chk_time */

/* the role of chk_time as controlling the update of tn through invocations of
inc_time(1)
is  made more apparent if we contrast it with the following variant:

proc chk_time: tn {
 todo("inc_time(1);");
}

which effectively updates the clock as fast as the eden queuing and processing speed
permits
*/

tnsecs is abs_time(tn);


/*ext0.d*/

%scout
# meta-interaction: action by the modeller to change the
# state of the computer model e.g. in order to make more
# space on the screen, by relocating image of clock and
# by shrinking the clock image.

%scout
mag = 1;
clockwinPos = {10,10};

%donald
# extension 0

# clock/t is a donald int variable
# representing time elapsed in minutes
# from midnight (1440 per day)

clock/t = 134
clock/t = 670

# semantics of adding time?
# should a design for a watch have a particular time on it?

# potentially represents aspect of watch state that is beyond user control
# - reflecting actual user agency over time in real-world context

%eden
include("autotime.e");
/* this maintains a variable tnsecs that records number of seconds
elapsed in real-time from an arbitrary initial time: demo by
proc wtnsecs: tnsecs { writeln(tnsecs); }
*/
```

```
_clock_t is abs_time(tn) / 60;/* integer division, so integer result */

/* user agency over whether watch is running, realised by "Interrupt"
on tkeden input interface (construe as e.g. "remove battery") and
touch(&new_time);
(which we can construe as e.g. "insert battery")
*/

/* also have two kinds of watch: those where operation is perceptible
(e.g. via audible tick) and those where operation indiscernible
proc wtnsecs {};
*/

%donald
clock/t = 0
# potentially represents a user of the watch - setting the time
# many possible reasons for setting the time (e.g. watch is fast,
# user is in new time zone, designer wants to view hands overlapping).

%eden
uk_time is tnsecs / 60;
japan_time is uk_time + 480;
_clock_t is japan_time;

%donald
# extension 1

# edge is the circle that defines the outer rim of the clock face
# adding inner circle can be construed as a designer's action

within clock {
circle inner_edge
real width_edge
inner_edge = circle(center, radius - width_edge)
 width_edge = 20.0

# aesthetic issue of where to place the inner edge

width_edge = 10.0
}

# note that such action is construed as changing the watch
# as opposed to changing the state of the watch

# extension 2

# adding a second hand: potentially a design decision, but could also
# be construed as taking account of a hitherto neglected observable

%donald
within clock {
line secHand
real secAngle, size_secHand
secHand = [center + {size_secHand*radius @ secAngle}, center]
size_secHand = 0.8
}

# no position for second hand yet, so not on display
# before we give proper definition of dependency on time, can experiment
# with arbitrary values (meta-interaction by the model constructor
```

```
clock/secAngle = pi

# also might introduce explicit observables for size_minHand etc here
within clock {
real size_minHand
size_minHand = 0.75
minHand = [center + {size_minHand * radius @ minAngle}, center]
}

# now specify the position of the second hand as function of time
%eden
sec_mod_60 is tnsecs % 60;
/* determine how many seconds elapsed since last whole minute time */

%donald
within clock {
secAngle = (pi div 2.0) - float (sec_mod_60!)* (pi div 30.0)
}

%eden
A_clock_secHand = "color=red";

/* construed as a design feature */

/* compare the following implausible associations of observables */

_clock_size_secHand is (float(sec_mod_60) / 60.0) * _clock_size_minHand;

%donald
# the minute hand comes loose
clock/minAngle = - pi div 2

# repair is difficult, unless have observed correct configuration first
```

Appendix B

Extracts of definitive scripts for timetabling model

by EM group members

```
/*timetable.e*/

include("util.e");

/* ============
   Input data
   ============ */

class = ['x','y','z'];
staff = ['X','Y'];
slots = [1,2,3,4,5,7,8,10];

AVSTAFF=[['X',[2,3,5]],['Y',[3,5,6]]];
SPSTAFF=[['X',['x','y']],['Y',['z','x']]];
avstud=[['x',[1,2,3]],['y',[2,3,4]],['z',[3,4,5]]];

TT=[['x',2],['y',3],['x',4]];
ASS=[['x','X'],['y','Y']];

/* ======================================
   functions that contribute to messages
   ====================================== */

func endmsg /* $1=student */
{
   return ".^. for student " // $1 // "\n";
}

func endMSG /* $1=staff */
{
   return ".^. for staff " // $1 // "\n";
}

func single_slot /* $1 = slx, $2 = student */
{
return ($1# != 1) ? $2 // " has " // tostr($1#) // " slots\n" : "";
}

func avail_x /* $1 = student, $2 = slx, $3 = avx */
{
return (meet($2,$3)!=$2) ?
  $1 // " unavailable"
// " slx is " // tostr($2)
// " avx is " // tostr($3) // "\n"
:
"";
}
```

```
func has_assessor /* $1 = student, $2 = asx */
{
auto n;
n = $2#;
return ((n>2) | | (n==0)) ?
$1 // " has " // tostr(n) // " assessors\n"
:
"";
}


func avail_X /* $1 = staffmem, $2 = SLX, $3 = AVX */
{
return (meet($2, $3) != $2) ?
  $1 // " unavailable"
// " SLX is " // tostr($2)
// " AVX is " // tostr($1) // "\n"
:
"";
}

func suit_X /* $1 = staffmem, $2 = ASX, $3 = SPX */
{
return (meet($2, $3) != $2) ?
  $1 // " unsuitable"
// " ASX = " // tostr($2)
// " SPX = " // tostr($3) // "\n"
:
"";
}

func count { return $1#; }

/* need to do something about it */

header =
"\n********\n" //
"TT is " // tostr(TT) // "\n" //
"ASS is " // tostr(ASS) // "\n" //
"AVSTAFF is " // tostr(AVSTAFF) // "\n" //
"avstud is " // tostr(avstud) // "\n" //
"SPSTAFF is " // tostr(SPSTAFF) // "\n" //
"*****\n";

func not_dblebk /* $1 = TT */
{
   auto i,j,l,overbook;
   overbook = "";
   for (i=1; i<=20; i++) {
l = [];
for (j=1; j<=$1#; j++) {
   if ($1[j][2]==i) {
append l, $1[j][1];
   }
}
if (l#>1) overbook = overbook // "Slot " // tostr(i) // " overbooked\n";
   }
   return overbook;
}
```

```
/* ======================
   elementary functions
   ====================== */

func meet
{
auto i,j,st;
st=[];
for (i=1; i<=($1)#; i++) {
for (j=1; j<=($2)#; j++) {
if ($1[i]==$2[j]) {
append st, $1[i];
}
}
}
return(st);
}

func proj2
{
auto i,l;
l = [];
for (i=1; i<=($1)#; i++) {
if (($1)[i][1]==$2) {append l, ($1)[i][2];}
}
return(l);
}

func proj2_1 { return proj2($1, $2)[1]; }

func proj1
{
auto i,l;
l = [];
for (i=1; i<=($1)#; i++) {
if (($1)[i][2]==$2) {append l, ($1)[i][1];}
}
return(l);
}

func join
{
auto i,j;
l=[];
for (i=1; i<=($1)#; i++) {
for (j=1; j<=($2)#; j++) {
if (($1)[i][1]==($2)[j][1]) {
append l, [($1)[i][2],($2)[j][2]];
}
}
}
return (l);
}
```

```
/* =============
   definitions
   ============= */

NPX is map(count, ASX);
asx is map(proj2, [ASS], class);
slx is map(proj2, [TT], class);
SLX is map(proj2, [join(ASS,TT)], staff);
ASX is map(proj1, [ASS], staff);
AVX is map(proj2_1, [AVSTAFF], staff);
SPX is map(proj2_1, [SPSTAFF], staff);
avx is map(proj2_1, [avstud], class);

e_student is map(endmsg, class);
e_staff is map(endMSG, staff);
s_s is combine(single_slot, [slx, class]);
a_x is combine(avail_x, [class, slx, avx]);
h_a is combine(has_assessor, [class, asx]);
a_X is combine(suit_X, [staff, SLX, AVX]);
s_X is combine(suit_X, [staff, ASX, SPX]);
msg is combine(strcat, [s_s, a_x, h_a, e_student]);
MSG is combine(strcat, [a_X, s_X, e_staff]);

/* =============
   show result
   ============= */

proc showheader : TT, ASS, AVSTAFF, avstud, SPSTAFF
{
writeln(header);
writeln(not_dblebk(TT));
}

proc showresult : class, staff
{
map(writeln, msg);
map(writeln, MSG);
}
```

```
/*timetable.s*/

%scout
window dataW;
string header;
dataW = {
frame:([{1.c, 1.r}, 8, 50]),
string:header,
border:1
};

window dbW;
string ndb;
dbW = {
frame:([{1.c, 10.r}, 5, 30]),
string:ndb,
border:1
};

window studentW;
string studmsg;
studentW = {
frame:([{1.c, 15.r}, 5, 30]),
string:studmsg,
border:1
};

window staffW;
string staffmsg;
staffW = {
frame:([{1.c, 20.r}, 5, 30]),
string:staffmsg,
border:1
};

screen = < dataW / dbW / studentW / staffW >;

integer student, staffmem;

%eden
include("timetable.e");

ndb is not_dblebk(TT);
studmsg is msg[locate(class, student)];
staffmsg is MSG[locate(staff, staffmem)];
func locate /* list, element */
{
auto i;
for (i = 1; $1[i] != $2; i++);
return i;
}

student = 'x';
staffmem = 'X';
%end
```

Appendix C

Definitive scripts for Restaurant Management Model

Developed by C. Roe and modified by T. Heron

```
/* NEWrest.e*/

/* Restaurant visualisation in Tkeden */
/* 27/1/00 by Chris Roe */
/*
 Revisions 19/11/01 by Tim Heron:
 Added blue line to show current time on timeline
Made colour highlighting hollow so we can see the text in the time line
Allow manager to choose table as well as suggesting a table (prevents manager double booking)
Reduced console commentary
Updated mouse events to work with latest tkeden
Now allows pre-defined events to take place instead of customers arriving randomly
Beeps when the clock stops
User not allowed to start clock until all events are dealt with

Future enhancements :
Combine tables together to create larger tables
Prevent user from restart clock without dealing with current events
Deal with multiple events occuring at the same time instance
*/

%eden

/* Restaurant definitions */

R_height is 70;
R_width is 80;
R_doorsize is 10;
R_numberoftables is 8;
R_tablecapacity is [4,4,2,2,4,4,2,2];
R_maxpartysize is 4;
R_tablewidth2 is 8;
R_tablewidth4 is 15;
R_tableheight2 is 10;
R_tableheight4 is 10;
R_busy is 50;
R_customersettlingtime is 2; /* number of minutes to hang up coats and sit down!! */
occupycolor = "green";
basecolor = "black";
starthour = 7;
stopClock = TRUE;
clock = 0;
potential_arrive_time = 0;
endclock = 240;
delay = 5000;
average_meal_time is 75;
booking_length is 90;
recorded_customer_data = []; /* to record information about customers */
customers_in_restaurant = [];
groups_in_restaurant = [];
allow_clock = 0; /* Counts events that need to be dealt with */
```

```
/* the fields in booking list are (name,table_number (determined by manager), number of
people, arrival time,cancelled) */
    b1name = "Russ";
    b1table = 1;
    b1nopeople = 4;
    b1arrivetime = 15;
    b1cancelled = FALSE;
    b2name = "Rasmequan";
    b2table = 7;
    b2nopeople = 2;
    b2arrivetime = 30;
    b2cancelled = FALSE;
    b3name = "Beynon";
    b3table = 4;
    b3nopeople = 2;
    b3arrivetime = 30;
    b3cancelled = FALSE;
    b4name = "Roe";
    b4table = 6;
    b4nopeople = 4;
    b4arrivetime = 45;
    b4cancelled = FALSE;
    b5name = "Ward";
    b5table = 8;
    b5nopeople = 2;
    b5arrivetime = 45;
    b5cancelled = FALSE;
    b6name = "Wong";
    b6table = 5;
    b6nopeople = 4;
    b6arrivetime = 60;
    b6cancelled = FALSE;
    b7name = "Maad";
    b7table = 3;
    b7nopeople = 2;
    b7arrivetime = 90;
    b7cancelled = FALSE;
    b8name = "Rungrattanaubol";
    b8table = 2;
    b8nopeople = 2;
    b8arrivetime = 90;
    b8cancelled = FALSE;

    booking_listis
[[b1name,b1table,b1nopeople,b1arrivetime,b1cancelled],[b2name,b2table,b2nopeople,b2arrivet
ime,b2cancelled],[b3name,b3table,b3nopeople,b3arrivetime,b3cancelled],[b4name,b4table,b4no
people,b4arrivetime,b4cancelled],[b5name,b5table,b5nopeople,b5arrivetime,b5cancelled],[b6na
me,b6table,b6nopeople,b6arrivetime,b6cancelled],[b7name,b7table,b7nopeople,b7arrivetime,b7
cancelled],[b8name,b8table,b8nopeople,b8arrivetime,b8cancelled]];

    seating_arrangement = [ [1,2,3,4] , [5,6,7,8], [9,10], [11,12], [13,14,15,16], [17,18,19,20], [21,22],
[23,24] ];
```

```
%donald

# This info is the DONALD script for the visualisation of the restaurant

viewport RESTVIEW

point R_bottomleft,R_topright,R_frontdoor,R_kitchendoor
line R_front_entrance,R_kitchen_entrance
rectangle R
R_bottomleft = {(100-R_width!) div 2,(100-R_height!) div 2}
R_topright = R_bottomleft+{R_width!,R_height!}
R = rectangle(R_bottomleft,R_topright)
R_frontdoor = R_topright-{0,R_height! div 2}
R_kitchendoor = R_bottomleft+{0,R_height! div 2}
R_front_entrance = [R_frontdoor-{0,R_doorsize! div 2},R_frontdoor+{0,R_doorsize! div 2}]
R_kitchen_entrance = [R_kitchendoor-{0,R_doorsize! div 2},R_kitchendoor+{0,R_doorsize!
div 2}]
?A_R_front_entrance is "color=black,linewidth=5";
?A_R_kitchen_entrance is "color=black,linewidth=5";

point
table1_topleft,table2_topleft,table3_topleft,table4_topleft,table5_topleft,table6_topleft,table7
_topleft,table8_topleft
rectangle
resttable1,resttable2,resttable3,resttable4,resttable5,resttable6,resttable7,resttable8
label
table1_label,table2_label,table3_label,table4_label,table5_label,table6_label,table7_label,table8_l
abel

table1_topleft = R_bottomleft+{10,10}
resttable1 = rectangle(table1_topleft,table1_topleft+{R_tablewidth4!,R_tableheight4!})
table1_label = label("Table1",table1_topleft+{R_tablewidth4! div 2,-15})
table2_topleft = R_bottomleft+{30,10}
resttable2 = rectangle(table2_topleft,table2_topleft+{R_tablewidth4!,R_tableheight4!})
table2_label = label("Table2",table2_topleft+{R_tablewidth4! div 2,-15})
table3_topleft = R_bottomleft+{50,10}
resttable3 = rectangle(table3_topleft,table3_topleft+{R_tablewidth2!,R_tableheight2!})
table3_label = label("Table3",table3_topleft+{R_tablewidth2! div 2,-15})
table4_topleft = R_bottomleft+{65,10}
resttable4 = rectangle(table4_topleft,table4_topleft+{R_tablewidth2!,R_tableheight2!})
table4_label = label("Table4",table4_topleft+{R_tablewidth2! div 2,-15})
table5_topleft = R_bottomleft+{10,R_height!-10-R_tableheight4!}
resttable5 = rectangle(table5_topleft,table5_topleft+{R_tablewidth4!,R_tableheight4!})
table5_label = label("Table5",table5_topleft+{R_tablewidth4! div 2,R_tableheight4!+15})
table6_topleft = R_bottomleft+{30,R_height!-10-R_tableheight4!}
resttable6 = rectangle(table6_topleft,table6_topleft+{R_tablewidth4!,R_tableheight4!})
table6_label = label("Table6",table6_topleft+{R_tablewidth4! div 2,R_tableheight4!+15})
table7_topleft = R_bottomleft+{50,R_height!-10-R_tableheight2!}
resttable7 = rectangle(table7_topleft,table7_topleft+{R_tablewidth2!,R_tableheight2!})
table7_label = label("Table7",table7_topleft+{R_tablewidth2! div 2,R_tableheight2!+15})
table8_topleft = R_bottomleft+{65,R_height!-10-R_tableheight2!}
resttable8 = rectangle(table8_topleft,table8_topleft+{R_tablewidth2!,R_tableheight2!})
table8_label = label("Table8",table8_topleft+{R_tablewidth2! div 2,R_tableheight2!+15})

# restaurant seats represented as circles which will be filled when occupied

point pseat1,pseat2,pseat3,pseat4 ,pseat5,pseat6,pseat7,pseat8, pseat9,pseat10,
pseat11,pseat12
point pseat13,pseat14,pseat15,pseat16 ,pseat17,pseat18,pseat19,pseat20 ,pseat21,pseat22
,pseat23,pseat24
```

```
circle seat1,seat2,seat3,seat4 ,seat5,seat6,seat7,seat8, seat9,seat10, seat11,seat12
circle seat13,seat14,seat15,seat16 ,seat17,seat18,seat19,seat20 ,seat21,seat22 ,seat23,seat24

pseat1 = table1_topleft+(3,-5)
pseat2 = table1_topleft+(R_tablewidth4!-3,-5)
pseat3 = table1_topleft+(3,R_tableheight4!+5)
pseat4 = table1_topleft+(R_tablewidth4!-3,R_tableheight4!+5)

pseat5 = table2_topleft+(3,-5)
pseat6 = table2_topleft+(R_tablewidth4!-3,-5)
pseat7 = table2_topleft+(3,R_tableheight4!+5)
pseat8 = table2_topleft+(R_tablewidth4!-3,R_tableheight4!+5)

pseat9 = table3_topleft+(R_tablewidth2! div 2,-5)
pseat10 = table3_topleft+(R_tablewidth2! div 2,R_tableheight2!+5)

pseat11 = table4_topleft+(R_tablewidth2! div 2,-5)
pseat12 = table4_topleft+(R_tablewidth2! div 2,R_tableheight2!+5)

pseat13 = table5_topleft+(3,-5)
pseat14 = table5_topleft+(R_tablewidth4!-3,-5)
pseat15 = table5_topleft+(3,R_tableheight4!+5)
pseat16 = table5_topleft+(R_tablewidth4!-3,R_tableheight4!+5)

pseat17 = table6_topleft+(3,-5)
pseat18 = table6_topleft+(R_tablewidth4!-3,-5)
pseat19 = table6_topleft+(3,R_tableheight4!+5)
pseat20 = table6_topleft+(R_tablewidth4!-3,R_tableheight4!+5)

pseat21 = table7_topleft+(R_tablewidth2! div 2,-5)
pseat22 = table7_topleft+(R_tablewidth2! div 2,R_tableheight2!+5)

pseat23 = table8_topleft+(R_tablewidth2! div 2,-5)
pseat24 = table8_topleft+(R_tablewidth2! div 2,R_tableheight2!+5)

seat1 = circle(pseat1,3)
seat2 = circle(pseat2,3)
seat3 = circle(pseat3,3)
seat4 = circle(pseat4,3)
seat5 = circle(pseat5,3)
seat6 = circle(pseat6,3)
seat7 = circle(pseat7,3)
seat8 = circle(pseat8,3)
seat9 = circle(pseat9,3)
seat10 = circle(pseat10,3)
seat11 = circle(pseat11,3)
seat12 = circle(pseat12,3)
seat13 = circle(pseat13,3)
seat14 = circle(pseat14,3)
seat15 = circle(pseat15,3)
seat16 = circle(pseat16,3)
seat17 = circle(pseat17,3)
seat18 = circle(pseat18,3)
seat19 = circle(pseat19,3)
seat20 = circle(pseat20,3)
seat21 = circle(pseat21,3)
seat22 = circle(pseat22,3)
seat23 = circle(pseat23,3)
seat24 = circle(pseat24,3)
```

```
%eden

include("NEWrestwindows.s");

%donald
viewport DONALD
%eden

proc occupyseats {
auto i;
 for (i=1; i<=$#;i++) {
  execute("A_seat"//str($[i])//" is \"color="//str(occupycolor)//",fill=solid\";");
 }
}

proc leaveseats {
auto i;

 for (i=1; i<=$#;i++) {
  execute("A_seat"//str($[i])//" is \"color="//str(basecolor)//",fill=hollow\";");
 }
}

func totime {
para c;
auto i,j;

i = c / 60;
j = c % 60;

ctime = strcat(i,":",j);

return ctime;
}

/* displays the current time in a window */

proc clocking : clock,stopClock
{
auto i;

 if (clock % 60 > 9)
 {clock_string = strcat("\n\n ",str(7+(clock / 60)),":",str(clock % 60));}
 else
 {clock_string = strcat("\n\n ",str(7+(clock / 60)),":0",str(clock % 60));}

 if (!stopClock)
 {potential_arrive_time = 0; todo("clock++;");}

 for (i=1; i<=delay; i++)
 {;}

 if (clock >= endclock)
 {todo("stopClock = TRUE;");}
}

   /* for customers entering the restaurant, if the time is correct by their booking (i.e in
booking_list)
     then they enter the restaurant and occupy their seats at a particular table */
```

```
proc customers_whohavebooked_entering : clock
{
auto i,j;

 for (i=1; i<=booking_list#; i++)
 {
  if ((booking_list[i][4]==clock)&&(booking_list[i][5]==FALSE)) /* i.e time to enter and not
cancelled*/
  {
   append customers_in_restaurant,booking_list[i];
   for (j=1; j<=booking_list[i][3]; j++)
   {
occupyseats(seating_arrangement[booking_list[i][2]][j]);
   }
  }
 }
}

/*
This function generates the amount of money that a group spends in the restaurant. It is this
function which is specified by the modeller at the moment which will generate the patterns in
data explorer. change this function to change the pattern of data for the restaurant. At the
moment this function depends on the number of people in the group perturbed by a small ran-
dom factor, and then takes account of the fluctuations spent by groups of differing sizes during
the evening. Aiming to get the following relationships, bigger groups spend more earlier,
smaller groups spend more later...
*/
/* currently takes no look at the amount of time in restaurant */

func moneyspent
{
 para timeinrestaurant,arrivetime,numberinparty;
 auto x,y,result;

 x = ((numberinparty*15)-10)+(random(numberinparty*8)-(numberinparty*4));

 if (numberinparty==2)
 {
  x = x + random(arrivetime/15)-random(arrivetime/30);
 }
 else
 {
  x = x + random((endclock-arrivetime)/15)-random((endclock-arrivetime)/30);
 }

 result = x;
 return result;
}

/* need to non-deterministically decide when customers are going to leave the restaurant */
proc customers_leaving : clock
{
 auto i,j,r;
 /* customers_in_restaurant will be of form [name, table_no, number of people, arrival time
*/
 if (customers_in_restaurant#>0)
 {
 i = 1;
 while (i<=customers_in_restaurant#)
 { /* since deleting elements - check inserted below for this */
```

```
    if (customers_in_restaurant==@) {break;}
                                        if        (((i<=customers_in_restaurant#)        &&
(customers_in_restaurant[i][4]+average_meal_time<clock)) | | (customers_in_restaurant[i][5]==
TRUE))
    {
     for (j=1; j<=customers_in_restaurant[i][3]; j++)
     {
  leaveseats(seating_arrangement[customers_in_restaurant[i][2]][j]);
     }
     writeln("Customer ",customers_in_restaurant[1]," is leaving the restaurant ");

     /* show the customer leaving the restaurant, manager can accept or ignore their data */
     _clwt = "Departure information";
     cleavewinnamefield_setText(strcat(str(customers_in_restaurant[i][1])," "));
     cleavewinnopeoplefield_setText(strcat(str(customers_in_restaurant[i][3])," "));
     cleavewinarrivetimefield_setText(strcat(str(customers_in_restaurant[i][4])," "));
     cleavewinallocatedtablefield_setText(strcat(str(customers_in_restaurant[i][2])," "));
     cleavewindeparttimefield_setText(strcat(str(clock)," "));
                              cleavewinmoneyspentfield_setText(strcat(str(moneyspent(clock-
customers_in_restaurant[i][4],customers_in_restaurant[i][4],customers_in_restaurant[i][3]))," 
"));

     allow_clock++;

     writeln(customers_in_restaurant,i);
     delete customers_in_restaurant,i;
     writeln(customers_in_restaurant);
     stopClock = TRUE;

    }
    i++;
   }
  }
 }


 /* draw timeline in the booking info window at the bottom */

 %donald

 viewport TTABLEDISPLAY

 #Utilised across viewports - hence shouldn't define in a viewport
 int blockheight,blockspacing
 blockspacing = 10
 blockheight      =      (trunc(ttableheight!)-40-(R_numberoftables!*blockspacing))      div
(R_numberoftables!)

 point tp,timeline_topleft
 label timelabel
 line currenttimevertline
 line bookingtimevertline
 rectangle timeline
 timeline_topleft = {50,10}
 tp = {25,20}
 timelabel = label("Time",tp)
 timeline = rectangle(timeline_topleft-{5,0},timeline_topleft+{ttablewidth!-100+5,20})
 ?A_timeline is "color=cyan,fill=solid";
```

```
    currenttimevertline=[timeline_topleft+{(clock!*(ttablewidth!-100))div(end-
clock!),0},timeline_topleft+{(clock!*(ttablewidth!-100))div(endclock!),20+(blockspac-
ing*R_numberoftables!)+(blockheight*R_numberoftables!)}]
    ?A_currenttimevertline is "color=red,linewidth=2";

    bookingtimevertline=[timeline_topleft+{(potential_arrive_time!*(ttablewidth!-100))div(end-
clock!),0},timeline_topleft+{(potential_arrive_time!*(ttablewidth!-100))div(end-
clock!),20+(blockspacing*R_numberoftables!)+(blockheight*R_numberoftables!)}]
    ?A_bookingtimevertline is "color=blue,linewidth=2";

    %eden

    /* because of the problems with donald declarations add the hours to the timeline using
execute and trigger each time the end
    time of the restaurant changes */

    proc puthours : endclock
    {
    auto i;

     for (i=0; i<=endclock / 60; i++)
     {
      execute("%donald
       viewport TTABLEDISPLAY
       point hour"//str(i+starthour)//"
       label hour"//str(i+starthour)//"label
         hour"//str(i+starthour)//" = {50+("//str(i)//"*((ttablewidth!-100) div (endclock! div
60))) ,20}
                hour"//str(i+starthour)//"label  =  label(\""//str(i+starthour)//"\",hour"//
str(i+starthour)//")
       ");
     }
    }

    proc showtablesintimetable : R_numberoftables
    {
    auto i,j;

     for (i=1; i<=R_numberoftables; i++)
     {
      execute("%donald
       viewport TTABLEDISPLAY
       point ptable"//str(i)//"
        rectangle table"//str(i)//"
       label table"//str(i)//"label
        ptable"//str(i)//"=timeline_topleft+{0,20+(blockspacing*"//str(i)//")+(blockheight*"/
/str(i-1)//")}
            table"//str(i)//"  =  rectangle(ptable"//str(i)//",ptable"//str(i)//"+{ttablewidth!-
100,blockheight})
        table"//str(i)//"label=label(\"Table"//str(i)//"\",tp+{0,20+(blockspacing*"//str(i)//
")+(blockheight*"//str(i-1)//")})
       ");
     }
    }

    proc showbookings : booking_list
    {
    auto i,j;

     for (i=1; i<=booking_list#; i++)
```

```
    {
     /* `id` turns the string id into an identifier */
     if (`"b"//str(i)//"cancelled"` == FALSE)
     {
      execute("%donald
       viewport TTABLEDISPLAY
       point pcust"//str(i)//"start,pcust"//str(i)//"end
       rectangle cust"//str(i)//"rect
       label cust"//str(i)//"label
         pcust"//str(i)//"start = timeline_topleft+{b"//str(i)//"arrivetime!*(ttablewidth!-100)
div endclock!,20+(blockspacing*b"//str(i)//"table!)+(blockheight*(b"//str(i)//"table!-1))}
       pcust"//str(i)//"end = pcust"//str(i)//"start + {booking_length!*(ttablewidth!-100) div
endclock!,blockheight}
        cust"//str(i)//"rect = rectangle(pcust"//str(i)//"start+{0,1},pcust"//str(i)//"end+{0,-
1})
          cust"//str(i)//"label = label(b"//str(i)//"name!,(pcust"//str(i)//"start+pcust"//
str(i)//"end) div 2)
      ");

      execute("%eden
       A_cust"//str(i)//"rect is
   ((_pcust"//str(i)//"start[2] < _timeline_topleft[2]+(clock*(ttablewidth-100))/endclock)
   && (_pcust"//str(i)//"end[2] > _timeline_topleft[2]+(clock*(ttablewidth-100))/endclock))
   ? \"outlinecolor=green\"
   : \"color=grey,fill=hollow\";

      A_cust"//str(i)//"label is
   ((_pcust"//str(i)//"start[2] < _timeline_topleft[2]+(clock*(ttablewidth-100))/endclock)
   && (_pcust"//str(i)//"end[2] > _timeline_topleft[2]+(clock*(ttablewidth-100))/endclock))
   ? \"color=green\"
   : \"color=black\";
    ");
    }
    else
    {
     execute("%donald
      ?A_cust"//str(i)//"rect is \"color=grey,outlinecolor=grey,fill=hollow\";
      ?A_cust"//str(i)//"label is \"color=grey\";
     ");
    }
   }
  }

  /* Now the routines for accepting/rejecting a new booking */

  func choosetable
  {
   para name,people,arrivetime;
   auto tableno,i,j;

   possibletables = [];
   for (i=1; i<=R_numberoftables; i++)
   {
   append possibletables,i;
   }

  /* check capacity of table with people in party */
   people = int(people); arrivetime=int(arrivetime);

   for (i=1; i<=R_numberoftables; i++) {
```

```
      /* writeln(R_tablecapacity[i]," ",people," ",possibletables[i],i); */
      if (R_tablecapacity[i] < people)
      {
    possibletables[i] = -1;
    /* writeln("fred"); */
    }
    else
    {
    /* writeln("wilma"); */
    }
     }

     /* writeln(possibletables); */

    /* now check we can fit their booking in where they want */
     for (i=1; i<=R_numberoftables; i++)
     {
      for (j=1; j<=booking_list#; j++)
      {
       if ((booking_list[j][2]==i)&&(booking_list[j][5]==FALSE)) /*i.e not cancelled */
       {
    if                                    ((arrivetime>booking_list[j][4]+booking_length) | |(arrive-
time+booking_length<booking_list[j][4]))
        {
      /* we're ok with this as it fits into the slot */
    } else {possibletables[i] = -1;}
       }
      }
     }

    /* possible tables now shows a list which includes some -1's. */
     writeln("Possible tables :",possibletables);
     resulttables = [];
     for (i=1; i<=R_numberoftables; i++)
     {
     if (possibletables[i]>0)
    {
    append resulttables,i;
    }
     }
     writeln("Result Tables :",resulttables);

    /* scope for a lot more improvement to the next bit! instead of being random should check
the amount of time it leaves */
    /* blank before or after the closest booking on that table */
     if (resulttables#!=0)
     {
      tableno = random(resulttables#)+1;
      writeln("Allocated table number ",resulttables[tableno]);
      return resulttables[tableno];

     }
     else
     {
     tableno=0;
    return tableno;
     }
    }
```

```
    Pname is "Anyone";
    Pnopeople is 1;
    Parrivetime is 0;
    Ptable is choosetable(Pname,Pnopeople,Parrivetime);

    proc acceptabooking : bookwinaccept_mouse_1
    {
     auto nextbooking,selectedtable;
     auto temp,temp2,temp3,resstr;

     if (bookwinaccept_mouse_1[2] != 4) { return; } /* Make sure we only accept the down click
*/

      _bwt = "New booking form";

      Pname = bookwinnamefield_getText();
      Pnopeople = bookwinnopeoplefield_getText();
      Parrivetime = bookwinarrivetimefield_getText();
      selectedtable = bookwinallocatedtablefield_getText();

      /* remove the new line characters */
      Pname = substr(Pname,1,Pname#-1);
      Pnopeople = substr(Pnopeople,1,Pnopeople#-1);
      Parrivetime = substr(Parrivetime,1,Parrivetime#-1);
      selectedtable = int(selectedtable);

      writeln("User Selected table is ",selectedtable);

      /*
      Make the following checks :
     That the selectedtable has the correct capacity
      That the selectedtable is available to be booked
      */

        if ((possibletables[selectedtable] > 0) && (selectedtable != 0) && (int(Pnopeople) <=
(R_tablecapacity[selectedtable])))
      {
       writeln("Booking made by ",Pname," for a party of ",Pnopeople, " at a time of ",Parrive-
time);

        /* Now have to decide which table to put it at and then put them there */
        nextbooking = booking_list#+1;
        execute("
         b"//str(nextbooking)//"name = \""//str(Pname)//"\";
         b"//str(nextbooking)//"table = "//str(selectedtable)//";
         b"//str(nextbooking)//"nopeople = "//str(Pnopeople)//";
         b"//str(nextbooking)//"arrivetime = "//str(Parrivetime)//";
         b"//str(nextbooking)//"cancelled = FALSE;
         temp = symboldetail(\"booking_list\")["//str(3)//"];
         temp2 = substr(temp,"//str(1)//",temp#-"//str(2)//");
                                                                    temp3          =
strcat(\",[b\",str(booking_list#+1),\"name,b\",str(booking_list#+1),\"table,b\",str(booking_li
st#+1),
            \"nopeople,b\",str(booking_list#+1),\"arrivetime,b\" ,str(booking_list#+1),\"can-
celled]]\" );
         resstr is strcat(temp2,temp3);
         writeln(\"a\");
         include(\"f.e\");
        ");
```

```
    bookwinnamefield_setText("");
    bookwinnopeoplefield_setText("");
    bookwinarrivetimefield_setText("");
    bookwinallocatedtablefield_setText("");

    allow_clock--;
    }
    else
    {
     writeln("No space can found in the restaurant.");
    }
  }

  proc rejectabooking : bookwinreject_mouse_1
  {
   auto x,y,z;

   if (bookwinreject_mouse_1[2] != 4) { return; } /* Make sure we only accept the down click
*/

   _bwt = "New booking form";
   writeln("Booking rejected by the manager.");
   x = bookwinnamefield_getText();
   y = bookwinnopeoplefield_getText();
   z = bookwinarrivetimefield_getText();

   writeln("Unreasonable demand made by ",x," for his party of ",y, " at a time of ",z);

   bookwinnamefield_setText("");
   bookwinnopeoplefield_setText("");
   bookwinarrivetimefield_setText("");
   bookwinallocatedtablefield_setText("");

   allow_clock--;
  }

  proc beep
  {
    writeln("\07");
  }

  proc clockstart : starttheclock_mouse_1
  {
    if (starttheclock_mouse_1[2] != 4) { return; } /* Make sure we only accept the down click */

    /* clock is only allowed to continue when zero events are left to deal with */
    if (allow_clock ==0)
    {
    stopClock = FALSE;
    }
    else
    {
    beep();
   writeln("Please deal with events.");

   /* HACK: Reduce the events by 1 so we can continue if we really want to */
   allow_clock--;
     }
  }
```

```
proc beep_on_stop : stopClock
{
if (stopClock == TRUE)
{
beep();
}
}

proc clockstop : stoptheclock_mouse_1
{
  if (stoptheclock_mouse_1[2] != 4) { return; } /* Make sure we only accept the down click */

  stopClock = TRUE;
}

/* To accept a cancellation we need to be able to uniquely identify the customer, so should
fill in as */
/* many of the fields required (name, party number, arrival time) */

proc acceptacancellation : cancelwinaccept_mouse_1
{
 auto name,numberinparty,arrivaltime;
 auto matchingcustomers,i;

 if (cancelwinaccept_mouse_1[2] != 4) { return; } /* Make sure we only accept the down click
*/

 name = cancelwinnamefield_getText();
 numberinparty = cancelwinnopeoplefield_getText();
 arrivaltime = cancelwinarrivetimefield_getText();
 name = substr(name,1,name#-1);
 numberinparty = substr(numberinparty,1,numberinparty#-1);
 arrivaltime = substr(arrivaltime,1,arrivaltime#-1);
 writeln(name," ",numberinparty," ",arrivaltime);
 writeln(booking_list#," = number of bookings");
 matchingcustomers = [];
 for (i=1;i<=booking_list#; i++)
 {
   writeln(booking_list[i][1],name,booking_list[i][3],numberinparty,booking_list[i][4],arriv-
altime);
  if (numberinparty==booking_list[i][3]) {writeln("A OK");}
     if  ((name==booking_list[i][1])&&(numberinparty==str(booking_list[i][3]))&&(arrival-
time==str(booking_list[i][4])))
  {
   append matchingcustomers,i;
  }
 }
 writeln(matchingcustomers);
 if (matchingcustomers#==0)
 {
  cancelwininfostring = "\nNo matching customers";
  writeln("No cancellation - incorrect data entered");
 }
 if (matchingcustomers#>1)
 {
  cancelwininfostring = "\nNo unique customer";
  writeln("No cancellation - incorrect data entered");
 }
   if  (matchingcustomers#==1)  {execute("b"//str(matchingcustomers[1])//"cancelled  =
TRUE;");}
```

```
    cancelwinnamefield_setText("");
    cancelwinnopeoplefield_setText("");
    cancelwinarrivetimefield_setText("");

    allow_clock--;

    _cwt = "Cancellation form";
  }

  proc customer_leaving_restaurant_usedata : cleavewinaccept_mouse_1
  {
    auto name,partysize,arrtime,tableno,deptime,moneyspent;

    if (cleavewinaccept_mouse_1[2] != 4) { return; } /* Make sure we only accept the down click
*/
    _clwt = "Departure information";

    name = cleavewinnamefield_getText();
    partysize = cleavewinnopeoplefield_getText();
    arrtime = cleavewinarrivetimefield_getText();
    tableno = cleavewinallocatedtablefield_getText();
    deptime = cleavewindeparttimefield_getText();
    moneyspent = cleavewinmoneyspentfield_getText();

    /* Remove trailing new-line character */
    name = substr(name,1,name#-1);
    partysize = substr(partysize,1,partysize#-1);
    arrtime = substr(arrtime,1,arrtime#-1);
    tableno = substr(tableno,1,tableno#-1);
    deptime = substr(deptime,1,deptime#-1);
    moneyspent = substr(moneyspent,1,moneyspent#-1);

      append recorded_customer_data,[name,partysize,tableno,arrtime,int(deptime)-int(arr-
time),moneyspent];

    /* Clear fields now they have been accepted */
    cleavewinnamefield_setText("");
    cleavewinnopeoplefield_setText("");
    cleavewinarrivetimefield_setText("");
    cleavewinallocatedtablefield_setText("");
    cleavewindeparttimefield_setText("");
    cleavewinmoneyspentfield_setText("");

    allow_clock--;
  }

  func random
  {
  para x;
  return rand(x)%x;
  }

  /*
  Pre-defined events
  Fields are :
  Time of event
  Type of event {ENTER,TELEPHONE,CANCEL}
  Name of booker
  Number of people
  Time to book          */
```

```
customer_events is [
[30,"ENTER","Joy","3","120"],
[40,"TELEPHONE","Russ","2","150"],
[50,"CANCEL","Russ","2","150"]
];

proc customer_enters_restaurant : clock
{
 /* Find events that should occur at this time */
 auto i,current_event;

 for (i=1; i<=customer_events#; i++)
 {
 current_event = customer_events[i];
if (current_event[1] == clock && current_event[2]=="ENTER")
{
_bwt = "Customer Enters Restaurant";
bookwinnamefield_setText(strcat(current_event[3]," "));
bookwinnopeoplefield_setText(strcat(current_event[4]," ")); /* add blank space cause this
needs chopping if entered into text box */
bookwinarrivetimefield_setText(strcat(current_event[5]," "));

Pname = current_event[3];
Pnopeople = current_event[4];
Parrivetime = current_event[5];
bookwinallocatedtablefield_setText(strcat(str(Ptable)," "));

allow_clock++;

potential_arrive_time=int(Parrivetime);

stopClock = TRUE;
}
 }
}

proc telephone_booking_restaurant : clock
{
 /* Find events that should occur at this time */
 auto i,current_event;

 for (i=1; i<=customer_events#; i++)
 {
 current_event = customer_events[i];
if (current_event[1] == clock && current_event[2]=="TELEPHONE")
{
_bwt = "Telephone Enquiry";
bookwinnamefield_setText(strcat(current_event[3]," "));
bookwinnopeoplefield_setText(strcat(current_event[4]," ")); /* add blank space cause this
needs chopping if entered into text box */
bookwinarrivetimefield_setText(strcat(current_event[5]," "));

Pname = current_event[3];
Pnopeople = current_event[4];
Parrivetime = current_event[5];
bookwinallocatedtablefield_setText(strcat(str(Ptable)," "));

potential_arrive_time=int(Parrivetime);

allow_clock++;
```

```
stopClock = TRUE;
}
 }
}

proc telephone_cancellation_restaurant : clock
{
 /* Find events that should occur at this time */
 auto i,current_event;

 for (i=1; i<=customer_events#; i++)
 {
 current_event = customer_events[i];
if (current_event[1] == clock && current_event[2]=="CANCEL")
{
_cwt = "Telephone Cancellation";
cancelwinnamefield_setText(strcat(current_event[3]," "));
cancelwinnopeoplefield_setText(strcat(current_event[4]," ")); /* add blank space cause this
needs chopping if entered into text box */
cancelwinarrivetimefield_setText(strcat(current_event[5]," "));

allow_clock++;

stopClock = TRUE;
}
 }
```

```
/*NEWrestwindows.s*/

%donald

# NOW put this restaurant into a scout window

viewport RESTTITLE
label resttitlestring
resttitlestring = label("View of restaurant",{restwidth! div 2,titleheight! div 2})

%scout

integer restwidth,restheight,titleheight,bw;
point base;
string rest_string,borderColor;
base = {20,50};
restwidth = 500;
restheight = 300;
titleheight = 15;
bw = 5;
borderColor = "Maroon";

window restTitleBar = {
      type: DONALD,
      pict : "RESTTITLE",
      box:[base-{0,15},base+{restwidth,0}],
      xmin :0,
      ymin :0,
      xmax :restwidth,
      ymax : titleheight,
      border: 1,
      fgcolor: "white",
      bgcolor: borderColor,
      sensitive: MOTION
};

window restdisp = {
            type: DONALD,
  pict : "RESTVIEW",
  box:[base,base+{restwidth,restheight}],
  xmin : 0,
  ymin : 0,
  xmax : 100,
  ymax : 100,
            bgcolor : "grey",
            border: 1,
            relief: "raised",
            sensitive : ON
};

window restBack = {
      type: TEXT,
      string: "",
 frame:([base-{bw,15+bw},base+{bw+restwidth,bw+restheight}]),
      bgcolor: borderColor,
      border: 1
};
```

```
%donald
viewport CLOCKTITLE
label clocktitlestring
clocktitlestring = label("Clock",{clockwidth! div 2,titleheight! div 2})

%scout

integer clockwidth,clockheight;
point base_2;
string clock_string;
base_2 = base+{restwidth+50,0};
clockwidth = 150;
clockheight = 50;

window clockTitleBar = {
     type: DONALD,
     pict : "CLOCKTITLE",
     box:[base_2-{0,15},base_2+{clockwidth,0}],
     xmin :0,
     ymin :0,
     xmax :clockwidth,
     ymax : titleheight,
     border: 1,
     fgcolor: "white",
     bgcolor: borderColor,
     sensitive: MOTION
};
window clockdisp = {
          type: TEXT,
 frame:([base_2,base_2+{clockwidth,clockheight}]),
          string : clock_string,
          bgcolor : "grey",
          border: 1,
          relief: "raised",
          sensitive : ON
};

window clockBack = {
     type: TEXT,
     string: "",
frame:([base_2-{bw,15+bw},base_2+{bw+clockwidth,bw+clockheight}]),
     bgcolor: borderColor,
     border: 1
};

point base_4;
base_4 = base_2 + {50,0};
string stoptheclockcolor,starttheclockcolor;

window stoptheclock = {
type : TEXT
string : "\nStop clock",
frame :([base_4,base_4+{100,clockheight/2}]),
bgcolor : stoptheclockcolor,
     border : 1,
sensitive : ON
};

point base_5;
base_5 = base_4+{0,clockheight/2};
```

```
window startttheclock = {
type : TEXT
string : "\nStart clock",
frame :([base_5,base_5+{100,clockheight/2}]),
bgcolor : startttheclockcolor,
    border : 1,
sensitive : ON
};

%eden
stoptheclockcolor is (stopClock==TRUE) ? "red" : DFbgcolor;
startttheclockcolor is (stopClock==FALSE) ? "green" : DFbgcolor;

%scout

# Booking timetable window

integer ttablewidth,ttableheight;
point base_3;
base_3 = {20,50+restheight+50};
ttablewidth = 500;
ttableheight = 300;

window ttabledisp = {
        type: DONALD,
        box:[base_3,base_3+{ttablewidth,ttableheight}],
        pict: "TTABLEDISPLAY",
        xmin: 0,
        xmax: ttablewidth,
        ymin: 0,
        ymax: ttableheight,
        bgcolor : "grey",
        bdcolor : borderColor,
        border: 1,
        relief: "raised",
        sensitive : MOTION
};

window ttableBack = {
    type: TEXT,
    string: "",
frame:([base_3-{bw,15+bw},base_3+{bw+ttablewidth,bw+ttableheight}]),
    bgcolor: borderColor,
    border: 1
};

%donald
viewport TTABLETITLE
label ttablestring
ttablestring = label("Booking information",{ttablewidth! div 2,titleheight! div 2})

%scout

window ttableTitleBar = {
    type: DONALD,
    pict : "TTABLETITLE",
    box:[base_3-{0,15},base_3+{ttablewidth,0}],
    xmin :0,
    ymin :0,
```

```
        xmax :ttablewidth,
        ymax : titleheight,
        border: 1,
        fgcolor: "white",
        bgcolor: borderColor,
        sensitive: MOTION
    };

    # New booking window

    integer bookwinwidth,bookwinheight;
    string bookwinfields;
    point base_6;
    base_6 = {20+restwidth+50,50+clockheight+50};
    bookwinwidth = 300;
    bookwinheight = 150;
    bookwinfields = "\n Name       : \n\n People     : \n\n Arrival time : \n\n Table number
:";

    window bookwindisp = {
                type: TEXT,
                frame:([base_6,base_6+{bookwinwidth,bookwinheight}]),
     string : bookwinfields,
                bgcolor : "grey",
                bdcolor : borderColor,
                border: 1,
                relief: "raised",
                sensitive : MOTION
    };

    window bookwinBack = {
        type: TEXT,
        string: "",
    frame:([base_6-{bw,15+bw},base_6+{bw+bookwinwidth,bw+bookwinheight}]),
        bgcolor: borderColor,
        border: 1
    };

    %donald
    viewport BOOKWINTITLE
    label bookwinstring
    char bwt
    bwt = "New booking form"
    bookwinstring = label(bwt,{bookwinwidth! div 2,bookwinheight! div 2})

    %scout

    window bookwinTitleBar = {
        type: DONALD,
        pict : "BOOKWINTITLE",
        box:[base_6-{0,15},base_6+{bookwinwidth,0}],
        xmin :0,
        ymin :0,
        xmax :bookwinwidth,
        ymax : bookwinheight,
        border: 1,
        fgcolor: "white",
        bgcolor: borderColor,
        sensitive: MOTION
    };
```

```
    string bookwinacceptcolor,bookwinrejectcolor;
    bookwinacceptcolor = "green";
    bookwinrejectcolor = "red";

    window bookwinaccept = {
    type : TEXT
    string : "\n Accept this booking",
    frame :([base_6+{0,120},base_6+{0,bookwinheight}+{bookwinwidth/2,0}]),
    bgcolor : bookwinacceptcolor,
        border : 1,
    sensitive : ON
    };

    window bookwinreject = {
    type : TEXT
    string : "\n Reject this booking",
    frame     :([base_6+{0,120}+{bookwinwidth      /2,0},base_6+{0,bookwinheight}+{bookwin-
width,0}]),
    bgcolor : bookwinrejectcolor,
        border : 1,
    sensitive : ON
    };

    window bookwinnamefield = {
    type : TEXTBOX
    frame : ([base_6+{120,10},{25,1}]),
    bgcolor : "grey",
    relief : "sunken",
    border : 1,
    sensitive : ON
    };
    window bookwinnopeoplefield = {
    type : TEXTBOX
    frame : ([base_6+{120,37},{25,1}]),
    bgcolor : "grey",
    relief : "sunken",
    border : 1,
    sensitive : ON
    };
    window bookwinarrivetimefield = {
    type : TEXTBOX
    frame : ([base_6+{120,64},{25,1}]),
    bgcolor : "grey",
    relief : "sunken",
    border : 1,
    sensitive : ON
    };

    window bookwinallocatedtablefield = {
    type : TEXTBOX
    frame : ([base_6+{120,91},{25,1}]),
    bgcolor : "DarkGrey",
    relief : "sunken",
    border : 1,
    sensitive : ON
    };
```

```
    # New cancellation window

    integer cancelwinwidth,cancelwinheight;
    string cancelwinfields;
    point base_7;
    base_7 = base_6+{0,50+bookwinheight};
    cancelwinwidth = 300;
    cancelwinheight = 150;
    cancelwinfields = "\n Name       : \n\n People     : \n\n Arrival time : \n\n Table number
:";

    window cancelwindisp = {
                type: TEXT,
                frame:([base_7,base_7+{cancelwinwidth,cancelwinheight}]),
     string : cancelwinfields,
                bgcolor : "grey",
                bdcolor : borderColor,
                border: 1,
                relief: "raised",
                sensitive : MOTION
    };

    window cancelwinBack = {
         type: TEXT,
         string: "",
    frame:([base_7-{bw,15+bw},base_7+{bw+cancelwinwidth,bw+cancelwinheight}]),
         bgcolor: borderColor,
         border: 1
    };

    %donald
    viewport CANCELWINTITLE
    label cancelwinstring
    char cwt
    cwt = "Cancellation form"
    cancelwinstring = label(cwt,{cancelwinwidth! div 2,cancelwinheight! div 2})

    %scout

    window cancelwinTitleBar = {
         type: DONALD,
         pict : "CANCELWINTITLE",
         box:[base_7-{0,15},base_7+{cancelwinwidth,0}],
         xmin :0,
         ymin :0,
         xmax :cancelwinwidth,
         ymax : cancelwinheight,
         border: 1,
         fgcolor: "white",
         bgcolor: borderColor,
         sensitive: MOTION
    };

    string cancelwinacceptcolor,cancelwininfostring;
    cancelwinacceptcolor = "green";
    cancelwininfostring = "";
```

```
    window cancelwinaccept = {
    type : TEXT
    string : "\n Accept cancellation",
    frame :([base_7+{0,120},base_7+{0,cancelwinheight}+{cancelwinwidth /2,0}]),
    alignment : CENTRE,
    bgcolor : cancelwinacceptcolor,
        border : 1,
    sensitive : ON
    };

    window cancelwininfo = {
    type : TEXT
    string : cancelwininfostring,
    frame  :([base_7+{cancelwinwidth /2,120},base_7+{cancelwinwidth / 2,cancelwin-
height}+{cancelwinwidth/2,0}]),
    alignment : CENTRE,
    bgcolor : "grey",
        border : 1,
    sensitive : ON
    };

    window cancelwinnamefield = {
    type : TEXTBOX
    frame : ([base_7+{120,10},{25,1}]),
    bgcolor : "grey",
    relief : "sunken",
    border : 1,
    sensitive : ON
    };
    window cancelwinnopeoplefield = {
    type : TEXTBOX
    frame : ([base_7+{120,37},{25,1}]),
    bgcolor : "grey",
    relief : "sunken",
    border : 1,
    sensitive : ON
    };
    window cancelwinarrivetimefield = {
    type : TEXTBOX
    frame : ([base_7+{120,64},{25,1}]),
    bgcolor : "grey",
    relief : "sunken",
    border : 1,
    sensitive : ON
    };

    window cancelwinallocatedtablefield = {
    type : TEXTBOX
    frame : ([base_7+{120,91},{25,1}]),
    bgcolor : "DarkGrey",
    relief : "sunken",
    border : 1,
    sensitive : ON
    };
```

```
    # Customer leaves restaurant window

    integer cleavewinwidth,cleavewinheight;
    string cleavewinfields;
    point base_8;
    base_8 = base_7+{0,50+cancelwinheight};
    cleavewinwidth = 300;
    cleavewinheight = 210;
    cleavewinfields = "\n Name      : \n\n People     : \n\n Arrival time : \n\n Table number
: \n\n Depart time  : \n\n Money spent  :";

    window cleavewindisp = {
              type: TEXT,
              frame:([base_8,base_8+{cleavewinwidth,cleavewinheight}]),
     string : cleavewinfields,
              bgcolor : "grey",
              bdcolor : borderColor,
              border: 1,
              relief: "raised",
              sensitive : MOTION
    };

    window cleavewinBack = {
         type: TEXT,
         string: "",
    frame:([base_8-{bw,15+bw},base_8+{bw+cleavewinwidth,bw+cleavewinheight}]),
         bgcolor: borderColor,
         border: 1
    };

    %donald
    viewport CLEAVEWINTITLE
    label cleavewinstring
    char clwt
    clwt = "Departure information"
    cleavewinstring = label(clwt,{cleavewinwidth! div 2,cleavewinheight! div 2})

    %scout

    window cleavewinTitleBar = {
         type: DONALD,
         pict : "CLEAVEWINTITLE",
         box:[base_8-{0,15},base_8+{cleavewinwidth,0}],
         xmin :0,
         ymin :0,
         xmax :cleavewinwidth,
         ymax : cleavewinheight,
         border: 1,
         fgcolor: "white",
         bgcolor: borderColor,
         sensitive: MOTION
    };

    string cleavewinacceptcolor,cleavewinrejectcolor;
    cleavewinacceptcolor = "green";
    cleavewinrejectcolor = "red";
```

```
window cleavewinaccept = {
type : TEXT
string : "\n Accept customer data",
frame :([base_8+{0,180},base_8+{0,cleavewinheight}+{cleavewinwidth /2,0}]),
alignment : CENTRE,
bgcolor : cleavewinacceptcolor,
    border : 1,
sensitive : ON
};

window cleavewinnamefield = {
type : TEXTBOX
frame : ([base_8+{120,10},{25,1}]),
bgcolor : "grey",
relief : "sunken",
border : 1,
sensitive : ON
};
window cleavewinnopeoplefield = {
type : TEXTBOX
frame : ([base_8+{120,37},{25,1}]),
bgcolor : "grey",
relief : "sunken",
border : 1,
sensitive : ON
};
window cleavewinarrivetimefield = {
type : TEXTBOX
frame : ([base_8+{120,64},{25,1}]),
bgcolor : "grey",
relief : "sunken",
border : 1,
sensitive : ON
};

window cleavewinallocatedtablefield = {
type : TEXTBOX
frame : ([base_8+{120,91},{25,1}]),
bgcolor : "grey",
relief : "sunken",
border : 1,
sensitive : ON
};

window cleavewindeparttimefield = {
type : TEXTBOX
frame : ([base_8+{120,118},{25,1}]),
bgcolor : "grey",
relief : "sunken",
border : 1,
sensitive : ON
};
window cleavewinmoneyspentfield = {
type : TEXTBOX
frame : ([base_8+{120,145},{25,1}]),
bgcolor : "grey",
relief : "sunken",
border : 1,
sensitive : ON
};
```

display timetablewindow = <ttableTitleBar/ttabledisp/ttableBack>;
display restaurantwindow = <restTitleBar/restdisp/restBack>;
display clockwindow = <clockTitleBar/stoptheclock/starttheclock/clockdisp/clockBack>;
display bookingwindow=<bookwinTitleBar/bookwinaccept/bookwinreject/bookwinnamefield/bookwinnopeoplefield/bookwinarrivetimefield/bookwinallocatedtablefield/bookwindisp/bookwinBack>;
display cancelwindow=<cancelwinTitleBar/cancelwinaccept/cancelwininfo/cancelwinnamefield/cancelwinnopeoplefield/cancelwinarrivetimefield/cancelwinallocatedtablefield/cancelwindisp/cancelwinBack>;
display customerleavewindow=<cleavewinTitleBar/cleavewinaccept/cleavewinnamefield/cleavewinnopeoplefield/cleavewinarrivetimefield/cleavewinallocatedtablefield/cleavewindeparttimefield/cleavewinmoneyspentfield/cleavewindisp/cleavewinBack>;
screen = restaurantwindow & timetablewindow & bookingwindow & cancelwindow & customerleavewindow & clockwindow;

Bibliography

[Abelson+85] H. Abelson, G. J. Sussman and J. Sussman. *Structure and interpretation of computer programs*. The MIT Press, 1985.

[Adam+98] F. Adam, M. Faby and C. Murphy. A framework for the classification of DSS usage across organizations. *Decision Support Systems: the International Journal*, Volume 22 (1998), Number 1, January 1998.

[Alter80] S. L. Alter. *Decision support systems: current practice and continuing challenges*. Addison-Wesley, 1980.

[Alter96] S. L. Alter. *Information systems: a management perspective*. 2nd Edition. Benjamin Cummings, 1996.

[Applegate+96] L. M. Applegate, W. F. McFarlan and J. L. Mckenney. *Corporate information systems management: the issues facing senior executive*. 4th Edition. Irwin, 1996.

[ArlingtonWeb] Arlington Software Corporation at http://www.arlingsoft. com /ergo.htm

[Avgerou+98] C. Avgerou and T. Cornford. *Developing information systems : concepts, issues and practice*. 2nd Edition. Macmillan, 1998.

[Avison92] D. E. Avison. *Information systems development: a database approach*. 2nd Edition. Blackwell Scientific Publications, 1992.

[Ayres99] R. Ayres. *The essence of professional issues in computing*. Prentice Hall, 1999.

[Baets98] W. R. J. Baets. *Organizational learning and knowledge technologies in a dynamic environment*. Kluwer, 1998.

[Barr+97] S. H. Barr and R. Sharda. Effectiveness of decision support systems: development or reliance effect. *Decision Support Systems: the International Journal*, Vol. 21, No. 2, October 1997.

[Berkeley49] E. C. Berkeley. *Giant brains or machines that think*. John Wiley & Sons, 1949.

[Beynon85] W. M. Beynon. Definitive notations for interaction. In *Proc. hci'85* Johnson and Cook (eds), Cambridge University Press, 1985.

[Beynon97a] W. M. Beynon. Introduction to definitive notations. Tutorial Materials for MSc module in Concurrent Programming, Department of Computer Science, University of Warwick, UK, 1997.

[Beynon97b] W. M. Beynon. Empirical Modelling for Educational Technology. In *Proc. of Cognitive Technology 1997*, pp.54-68, University of Aizu, Japan, 1997.

[Beynon98] W. M. Beynon. Empirical Modelling and the foundation of artificial intelligence. In C. Nehaniv (ed.), *Proc. of International Workshop on Computation, Metaphor, Analogy, and Agents*, University of Aizu, Japan, April 1998.

[Beynon+94] W. M. Beynon and M. S. Joy. Computer Programming for Noughts-and-Crosses: New Frontiers. In *Proc. PPIG'94*. 1994.

[Beynon+95a] W. M. Beynon, A. Cartwright, and Y.P. Yung. Databases from an agent-oriented perspective. Research report RR-278, Department of Computer Science, University of Warwick, 1995.

[Beynon+95b] W. M. Beynon and R. Cartwright. Empirical Modelling principles for cognitive artefacts. In *Proc. IEE Colloquium Design Systems with Users in Mind*, Digest No. 95/231, December, 1995.

[Beynon+95c] W. M. Beynon, P. E. Ness and S. B. Russ. Worlds before and beyond words. In *Proc.VF 1995*, Warwick University, 1995.

[Beynon+98] W. M. Beynon, R. Cartwright, J. Rungrattanaubol and P. H. Sun. Interactive situation models for systems development. Research report RR-353, Department of Computer Science, University of Warwick, UK, 1998.

[Beynon+00a] W. M. Beynon, S. Rasmequan, and S. Russ. The use of interactive situation models for the development of business solutions. In *Proc. of the Workshop on Business Information Research*, University of Rostock, Rostock, Germany, March 2000.

[Beynon+00b] W. M. Beynon, A. Ward, S. Maad, A. Wong, S. Rasmequan and S. Russ. The temposcope: a computer instrument for the idealist timetabler. Research Report 372, Department of Computer Science, University of Warwick, 2000.

[Beynon+01a] W. M. Beynon, Y-C Chen, H-W Hseu, S. Maad, S. Rasmequan, C. Roe, J. Rungrattanaubol, S. Russ, A. Ward and A. Wong. The computer as instrument. In [Beynon+01b].

[Beynon+01b] W. M. Beynon, C. L. Nehaniv and K. Dautenhahn (eds). Cognitive technology: instruments of mind. In *Proc. of the 4th International Conference*, CT2001, Coventry, UK, August 2001, Springer LNAI 2117, 2001.

[Beynon+02] W. M. Beynon, S. Rasmequan, and S. Russ. A new paradigm for computer-based decision support. To appear in a special issue of *Decision Support Systems: the International Journal*, vol. 33 issue 2, pp. 127 - 142, 2002.

[Bhargava+95a] H. K. Bhargava, A. S. King and D. S. McQuay. DecisionNet: modeling and decision support over the world wide web. In *Proc. of the third (3rd) ISDSS Conference*, Hong Kong, June 1995, at http://dnet.sm.nps.navy.mil

[Bhargava+95b] H. K. Bhargava, R. Krishnan and R. Muller. Sharing decision technologies over a global network. In *Proc. of the International Conference on Automation* (Indore, India), December 1995 at http://dnet.sm.nps.navy.mil

[Bhaskar+82] K. Bhaskar, P. Pope and R. Morris. Financial modelling with computers: a guide for management. The *Economist Intelligence Unit (EIU)*, Special Report No. 120, 1982.

[Boden90] M. A. Boden (ed). *The philosophy of artificial intelligence*. Oxford University Press, 1990.

[Bodily85] S. E. Bodily. *Modern decision making: a guide to modeling with decision support systems*. McGraw-Hill, 1985.

[Boud+85] D. Boud, R. Keogh and D. Walker (editors). *Reflection: turning experience into learning*, Kogan Page, 1985.

[Boutell68] W. S. Boutell. *Computer-oriented business systems.* Prentice Hall, 1968.

[Brooks75] F. P. Brooks. *The Mythical Man-Month: essays on software engineering,* Addison-Wesley, 1995 (First published in 1975, reprinted in 1983 and 1995).

[Brooks87] F. P. Brooks. No silver bullet: essence and accidents of software engineering. *IEEE Computer,* Vol. 20, No. 4, pp. 10 -19, 1987.

[Brookshear97] J. G. Brookshear. *Computer science: an overview.* Addison Wesley Longman, 1997.

[Buchanan+82] B. G. Buchanan and E. A. Feigenbaum. Foreward. In R. Davis and D. B. Lenat (eds), *Knowledge-based systems in artificial intelligence.* McGraw-Hill, 1982.

[Budde+92] R. Budde, K. Kautz, K. Kuhlenkamp and H. Zullighoven. *Prototyping: an approach to evolutionary system development.* Spinger-Verlag, 1992.

[Bundy83] A. Bundy. *The computer modelling of mathematical reasoning,* Academic Press, 1983.

[Burns+90] A. Burns and A. J. Wellings. *Real-time systems and their programming languages.* Addison-Wesley, 1990.

[Cartwright98] R. I. Cartwright. *Geometric aspects of Empirical Modelling: issues in design and implementation.* Ph.D. Thesis, Department of Computer Science, University of Warwick, UK, 1998.

[Cawkell93] A. E. Cawkell. Information management – degree of success. *Encyclopaedic Dictionary of Information Technology and Systems,* Bowker-Saur, 1993.

[Chaitin99] G. J. Chaitin. *The unknowable.* Springer, 1999.

[Charatan+01] Q. Charatan and A. Kans. *Java: the first semester.* McGraw Hill, 2001.

[Checkland81] P. Checkland. *Systems thinking, system practice.* John Wiley & Sons, 1981.

[Checkland89] P. Checkland. Soft system methodology. *IOS Human Systems Management*, Vol. 8, pages 273 -289, 1989.

[Checkland+90] P. Checkland and J. Scholes. *Soft systems methodology in action.* John Wiley & Sons, 1990.

[Checkland+98] P. Checkland and S. I. Holwell. *Information, systems and information systems: making sense of the field.* John Wiley & Sons, 1998.

[Corder89] C. Corder. *Taming your company computer.* McGraw Hill, 1989.

[Cortada96] J. Cortada. *A bibliographic guide to the history of computer applications (1950 - 1990).* Greenwood Press, 1996.

[Crane95] T. Crane. *The mechanical mind.* Penguin Books, 1995.

[CriticaltoolsWeb] http://www.criticaltools.com/index.html

[Cross67] H. Cross. A general management view of computers. *Computers and management: the 1967 Leatherbec lectures.* Harvard University, 1967.

[Crowe+96] M. Crowe, R. Beeby and J. Gammack. *Constructing systems and information: a process view.* McGraw-Hill, 1996.

[Czerniawska+98] F. Czerniawska and G. Potter. *Business in a virtual world: exploiting information for competitive advantage.* MacMillan, 1998.

[Davies95] S. R. Davies. *Spreadsheets in structural design.* Longman, 1995.

[Davies+91] L. J. Davies and P. W. J. Ledington. *Information in action: soft system methodology.* Macmillan Education, 1991.

[Davis+82] R. Davis and D. B. Lenat. *Knowledge-based systems in artificial intelligence.* McGraw-Hill, 1982.

[Deutschen+96] N. R. Deutschen, E. J. Bowers and J. W. Lankford. ASCC: The impact of a silver bullet. *AT&T Technical Journal*, Vol. 75, No. 1, January/February 1996.

[Dhar+97] V. Dhar and R. Stein. *Seven methods for transforming corporate data into business intelligence.* Prentice-Hall, 1997.

[Dromey82] R. G. Dromey. *How to solve it by computer.* Prentice-Hall, 1982.

[Edwards+97] J. S. Edwards and P. N. Finlay. *Decision making with computers: the spreadsheet and beyond.* Pitman Publishing, 1997.

[Elgood93] C. Elgood. *Hand book of management games.* 5th Edition. Gower Press, 1993.

[EMWeb] Empirical Modelling Web Site – http://www.dcs.warwick.ac.uk/modelling

[Etor86] J. R. Etor. *Spreadsheet computing in business education: an experimental approach.* Longman, 1986.

[Feigenbaum+95] E. A. Feigenbaum and J. Feldman. *Computers & thought.* American Association for Artificial Intelligence, 1995.

[Fick+80] G. Fick and R. H. Jr. Sprague (eds). *Decision support systems: issues & challenges.* Pergamon Press, 1980.

[Finlay89] P. Finlay. *Introducing decision support systems.* NCC Blackwell, 1989 (reprinted in 1994).

[Flynn98] D. Flynn. *Information systems requirements: determination & analysis.* McGraw-Hill, 1998.

[Forsyth89] R. Forsyth (ed). *Expert systems: principles and case studies.* 2nd Edition. Chapman and Hall Computing, 1989.

[French96] C. S. French. *Data processing and information technology.* 10th Edition. DP Publications, 1996.

[Galliers94] R. Galliers. *Information system research: issues, methods and practical guidelines.* Alfred Waller, 1994.

[Gilbert+99] N. Gilbert and K. G. Troitzsch. *Simulation for the social scientist.* Open University Press, 1999.

[Ginzberg+82] M. J. Ginzberg, W. Reitman, E. A. Stohr (eds). *Decision support systems.* North Holland, 1982.

[Godfrey93] M. D. Godfrey. First draft of a report on the EDVAC: John Von Neumann. *IEEE Annals of the History of Computing*, Vol. 15, No. 4, 1993.

[Goffman61] E. Goffman. Role distance (originally published in Encounters: Two studies in the sociology of interaction, 1961) . In C. Lemert and A. Branaman (eds), The self and social roles, *The Goffman reader*. Blackwell, 1997 (reprinted 2001).

[Gooding90] D. Gooding. *Experiment and the making of meaning*. Kluwer, 1990.

[Gooding01] D. Gooding. Experiment as an instrument of innovation: experience and embodied thought. In [Beynon+01b].

[Gorry+71] G. A. Gorry and M. S. Scott-Morton. A framework for management information systems. *Sloan Management Review*, 1971.

[Gosling89] P. E. Gosling. *Mastering spreadsheets*. Macmillan Education, 1989.

[Graham+88] I. Graham and P. L. Jones. *Expert systems: knowledge, uncertainty and decision*. Chapman and Hall Computing, 1988.

[Gray94] P. Gray (ed). *Decision support and executive information systems*. Prentice-Hall, 1994.

[Harel92] D. Harel. Biting the silver bullet: towards a brighter future for systems development. *IEEE Computer*, January 1992.

[Higginson65] V. M. Higginson. Managing with EDP: a look at the state of the art. AMA Research Study, American Management Association, 1965.

[Hirschheim+95] R. Hirschheim, H. K. Klein and K. Lyytinen. *Information systems development and data modeling: conceptual and philosophical foundations*. Cambridge University Press, 1995.

[Hornby+74] A. S. Hornby, A.P. Cowie and A.C. Gimson. *Oxford advanced learner's dictionary of current english*. Oxford University Press, 1974.

[Huber82] Huber. In G. R. Ungson and D. N. Braunstein (eds). *Decision making: an interdisciplinary inquiry*. Kent Publication, 1982.

[Humphreys+96] P. Humphreys, L. Bannon, A. McCosh, P. Migliarese and J. Pomerol (eds). *Implementing systems for supporting management decisions: concepts, methods and experiences.* Chapman & Hall, 1996.

[Isakowitz+95] T. Isakowitz, S. Schocken and Jr. H. C. Lucas. Toward a logical/ physical theory of spreadsheet modeling. *ACM Transactions on Information Systems,* Vol. 13, No. 1, pp. 1 - 37, January 1995.

[James12] W. James. *Essays in radical empiricism.* Bison Books, 1996 (first published in 1912).

[Jelassi+92] T. Jelassi, M. R. Klein and W. M. Mayon-White (eds). *Decision support systems: experiences and expectations.* North-Holland, 1992.

[Johnson-Laird93] P. Johnson-Laird. *The computer and the mind.* 2nd Edition. Fontana Press, 1993 (first published in 1983).

[Jordan68] N. Jordan. *Themes in speculative psychology.* Tavistock Publication, 1968.

[Kaplan86] S. J. Kaplan. Some future trends in spreadsheets. *LOTUS.* February 1986.

[Keen+78] P. G. W. Keen and M. S. S. Morton. *Decision support systems: an organizational perspective.* Addison-Wesley, 1978 .

[Kent78] W. Kent. *Data and reality: basic assumptions in data processing.* North-Holland, 1978.

[Kim+97] J. Kim and J. F. Lerch. Why is programming (sometimes) so difficult ?: programming as scientific discovery in multiple problem spaces. *Information Systems Research,* Vol. 8, No.1, March 1997.

[Klein+95] M. R. Klein and L. B. Methlie. *Knowledge-based decision support systems: with applications in business.* 2nd Edition. John Wiley & Sons, 1995.

[Kuhn70] T. S. Kuhn. *The structure of scientific revolutions.* University of Chicago Press, 1970.

[Land85] F. Land. Is an information theory enough ?. *The computer journal,* Vol. 28, No. 3, pp. 211 - 214, Oxford University Press, 1985.

[Lyytinen97] K. Lyytinen. Different perspectives on information systems: problems and solutions. *ACM Computing Surveys*, Vol.19, No.1, March 1997.

[Macro+87] A. Macro and J. Buxton. *The craft of software engineering*. Addison-Wesley, 1987.

[McDaniel94] G. McDaniel. *IBM dictionary of computing*. McGraw-Hill, 1994.

[McLean+86] E. R. McLean and H. G. Sol (eds). *Decision support systems: a decade in perspective*. North-Holland, 1986.

[Miller56] G. A. Miller. The magical number seven, plus or minus two. *Psychological Review*, 63, 81-97, 1956 (cited in P. Johnson-Laird, The computer and the mind. 2nd Edition, Fontana Press, 1988 .)

[Miller+88] K. R. Miller. *Artificial intelligence application for business management*. 2nd Edition. SEAI Technical Publication, 1988.

[Minsky88] M. Minsky. *The society of mind*. Picador, 1988.

[Mintzberg73] H. Mintzberg. *The nature of managerial work*. Haper & Row, 1973.

[Mintzberg91] H. Mintzberg. Crafting strategy. In The state of strategy, *Harvard Business Review*, 1985 - 1991.

[Nagel93] S. S. Nagel (ed). *Computer-aided decision analysis: theory and applications*. Quorum Books, 1993.

[Nardi93] B. A. Nardi. *A small matter of programming: perspectives on end user programming*. MIT Press, 1993.

[Ness97] P. E. Ness. *Creative software development: An Empirical Modelling framework*, PhD Thesis, Department of Computer Science, University of Warwick, UK, 1997.

[Neumann+53] J. Von Neumann and Oskar Morgen. *Theory of games and economic behavior*. 3rd Edition. Princeton, Princeton University Press, 1953.

[Nissen+91] H-E Nissen, H. K. Klein and R. Hirschheim. Information systems research: contemporary approaches & emergent traditions. *IFIP*, North-Holland, 1991.

[Norman83] D. A. Norman. Some observations on mental models. In D. Gentner and A. L. Stevens (eds) *Mental Models*. Lawrence Erlbaum Associates, 1983.

[Norman91] D. A. Norman. Cognitive Artifacts. In J. M. Carroll (ed) *Designing interaction: psychology at the human-computer interface*, Cambridge University Press, 1991.

[Norman98] D. A. Norman. *The invisible computer: why good products can fail.* MIT Press, 1998.

[Parton64] K. C. Parton. *The digital computer.* Pergamon Press, 1964.

[Payne99] S. J. Payne. Everyday problem solving. *International Encyclopedia of the Social & Behavioral Sciences*. Elsevier Science, 2001.

[Pidd96] M. Pidd. *Tools for thinking: modelling in management science*. Wiley, 1996.

[Polanyi67] M. Polanyi. *The tacit dimension.* Routledge and Kegan Paul, 1967.

[Polanyi83] M. Polanyi. *The tacit dimension.* Gloucester Mass, 1983.

[Polanyi+75] M. Polanyi and H. Prosch. *Meaning*. The University of Chicago Press, 1975.

[PowerWeb] D. J. Power. A brief history of spreadsheets. At http://www.dss-resources.com/history/sshistory.html.

[Price97] S. M. Price. Facilities planning: a perspective of an information age. *IIE Solution*, August, 1997.

[Ralston+00] A. Ralston, E. D. Reilly and D. Hemmendinger (eds). *Encyclopedia of Computer Science*. 4th Edition. Nature Publishing Group, 2000.

[Rasmequan+00a] S. Rasmequan, C. Roe, and S. Russ. Strategic decision support systems: an experience-based approach. In *Proc. of the 18th IASTED Conference on Applied Informatics*, 14-17 February, Innsbruck, Austria, 2000.

[Rasmequan+00b] S. Rasmequan and S. Russ. Cognitive artefacts for decision support. In *Proc. of the 2000 IEEE International Conference on Systems, Man & Cybernetics*, 8-11 October, Tennessee, USA, 2000.

[Reid61] L. A. Reid. *Ways of knowledge and experience.* George Allen & Unwin, 1961.

[Reuber97] R. Reuber. Management experience and management expertise. *Decision Support Systems: the International Journal*, Vol. 21, No. 2, October 1997.

[Rivett94] P. Rivett. *The craft of decision modelling.* John Wiley & Sons, 1994.

[Robson97] W. Robson. *Strategic management and information systems: an integrated approach.* 2nd Edition. Pitman Publishing, 1997.

[Robson+95] A. Robson and J. Pemberton. Designing user friendly models. *Spreadsheet User*, Vol. 2, No. 1, May 1995.

[Ronen+89] B. Ronen, M. A. Palley and H. C. Lucus (Jr.). Spreadsheet analysis and design. *Communications of the ACM*, Vol. 32, No. 1, January 1989.

[Rothery90] A. Rothery. *Modelling with spreadsheets.* Chartwell-Bratt, 1990.

[Rumelhart+88] D. E. Rumelhart and D. A. Norman. Representation in memory. In R. C. Atkinson, R. J. Herrnstein, G. Lindzey and R. D. Luce (eds). Stevens' handbook of experimental psychology. 2nd Edition, Vol. 2. John Wiley & Sons, 1988.

[Russ97] S. B. Russ. Empirical Modelling: the computer as a modelling medium. *Computer Bulletin*, April 1997.

[Russell79] B. Russell. *The problems of philosophy.* Oxford University Press, 1979 (first published in 1912).

[Sauter97] V. L. Sauter. *Decision support systems: an applied managerial approach.* John Wiley& Sons, 1997.

[Savage98] J. E. Savage. *Models of computation: exploring the power of computing.* Addison-Wesley, 1998.

[Searle92] J. R. Searle. *Minds, brains and science: the 1984 Reith lectures.* Penguin, 1992.

[Schlosser92] M. Schlosser. *Corporate finance: a model building approach.* 2nd Edition. Prentice Hall, 1992.

[Simon60] H. A. Simon. *The new science of management decision.* Harper & Row, 1960.

[Sol83] H. G. Sol (ed). *Processes and tools for decision support.* North Holland, 1983.

[Sperschneider+91] V. Sperschneider and G. Antoniou. *Logic a foundation for computer science.* Addison-Wesley, 1991.

[Sprague+82] R. H. Sprague and E. D. Carlson. *Building effective decision support systems.* Prentice-Hall, 1982.

[Sprague+93] R. H. Sprague and H. J. Watson (eds). *Decision support systems : putting theory into practice.* 3rd Edition. Prentice Hall, 1993.

[Sprague+96] R. H. Sprague and H. J. Watson (eds). *Decision support for management.* Prentice-Hall, 1996.

[Stevens+00] P. Stevens and R. Pooley. *Using UML: software engineering with objects and components.* Pearson Education, 2000.

[Stewart75] I. Stewart. *Concepts of modern mathematics.* Dover Publications, 1995 (first published 1975, reprinted 1981, 1995).

[Sullivan+91] J. W. Sullivan and S. W. Tyler. *Intelligent user interfaces.* ACM Press, 1991.

[Sun99] P. Sun. *Distributed Empirical Modelling and its application to software system development.* Ph.D. Thesis, Department of Computer Science, University of Warwick, UK, 1999.

[Sun+99a] P. Sun and W. M. Beynon. Empirical Modelling: a new approach to understanding requirements. In *Proc. of the 11th International Conference on Software Engineering and its Application,* Vol. 3, Paris, 1999.

[Sun+99b] P-H Sun, Y-C Chen, S. B. Russ, W. M. Beynon. Cultivating requirements in a situated requirements engineering process. Research report 357, Department of Computer Science, University of Warwick, UK, 1999.

[Sutherland98] J. W. Sutherland. *Towards a strategic management and decision technology.* Kluwer, 1998.

[Tanik+91] M. M. Tanik and E. S. Chan. *Fundamentals of computing for software engineers.* Van Nostrand Reinhold, 1991.

[Todd76] S. J. P. Todd. The Peterlee relational test vehicle. *IBM System Journal.* Vol. 15 (4), pp. 285-308, 1976.

[Tricker93] B. Tricker. *Harnessing information power.* Hong Kong University Press, 1993.

[Trigg85] R. Trigg. *Understanding social science: a philosophical introduction to the social science.* Blackwell, 1985.

[Trigg99] R. Trigg. *Rationality & science: can science explain everything ?.* Blackwell, 1999.

[Turban95] E. Turban. *Decision support and expert systems: management support systems.* 4th Edition. Prentice-Hall, 1995.

[Turban+98] E. Turban and J. Aronson. *Decision support systems and intelligent systems.* 5th Edition. Prentice-Hall, 1998.

[Turner96] M. Turner. *The Literary Mind.* Oxford University Press, 1996.

[Tuzhilin97] A. Tuzhilin. Editor's introduction to the special issue on knowledge discovery and its applications to business decision-making. *Decision Support Systems: the International Journal,* Vol. 21, No. 2, October 1997.

[Vliet93] H. V. Vliet. *Software design, software engineering: principles and practice.* John Wiley & Son, 1993.

[Waern89] Y. Waern. *Cognitive aspects of computer supported tasks.* John Wiley & Son, 1989.

[Warnock62] G. J. Warnock (ed). *Treatise concerning the principles of human knowledge*. Fontana, 1962.

[Way94] E. C. Way. *Knowledge representation and metaphor*. Intellect Books, 1994.

[Welbank83] M. Welbank. A review of knowledge acquisition techniques for expert systems. British Telecommunications. 1983.

[Whitehead61] A. N. Whitehead. *An introduction to mathematics*. Oxford University Press, 1961.

[Wilson84] B. Wilson. *Systems: Concepts, methodologies and application (application of SSM)*. Wiley, 1984 (first published in 1984, reprinted 1990).

[Wilson+01] L.B. Wilson and R. G. Clark. *Comparative programming languages*. 3th Edition. Addison-Wesley, 2001.

[Winograd+86] T. Winograd and F. Flores. *Understanding computers and cognition*. Addison-Wesley, 1986.

[Wyvill74] G. Wyvill. Pictorial description language. In *Proc. of DEC Users Society*, Zurich, Switzerland, 1974.

[Xia98] F. Xia. What's wrong with Software Engineering. Research methodology, *Software Engineering Notes*, ACM SIGSOFT, Vol. 23, No.1, pp. 62 -65, January 1998.

[Yourdon89] E. Yourdon. *Structured walk-throughs*. 4th Edition. Prentice-Hall, 1989.

[Yung88] Y. W. Yung. *The EDEN handbook*. University of Warwick, 1988.

[Yung92] Y. P. Yung. *Definitive programming: a paradigm for exploratory programming*. PhD thesis, Department of Computer Science, University of Warwick, 1992.